

80186/80188

Emulator

Users and Installation Manual

Serial No. B020300 and up

80186/80188 Emulator

Users and Installation Manual

This manual supports the
following TEKTRONIX products:

8300E38
8300E38 Option 01
8300P45
8300P46

Serial No. B020300 and up

*Please check for change information
at the rear of this manual*

Copyright © 1984 by Tektronix, Inc. All rights reserved.
Contents of this publication may not be reproduced in any
form without the permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered
by U.S. and foreign patents and/or pending patents.

TEKTRONIX, TEK, SCOPE-MOBILE, and  are regis-
tered trademarks of Tektronix, Inc. TELEQUIPMENT is a
registered trademark of Tektronix U.K. Limited.

There is no implied warranty of fitness for a particular pur-
pose. Tektronix, Inc. is not liable for consequential
damages.

Printed in U.S.A. Specification and price change privileges
are reserved.

ABOUT THIS MANUAL

GENERAL INFORMATION

This manual tells how to operate and install the 80186/80188 Emulator in a Tektronix 8500 Series development system. This manual also explains the features of the 8550 and 8540 development systems that are unique to the 80186 Emulator and 80188 Emulator.

Manual Application

This manual applies to 80186/80188 Emulators with serial numbers B020300 and up.

Manual Organization

This manual is divided into seven sections:

- Section 1** Contains general information about the emulator and the prototype control probes.
- Section 2** Contains user information specifically for the 80186 and 80188 Emulators.
- Section 3** Contains an emulator demonstration program.
- Section 4** Contains technical information related to the emulator.
- Section 5** Defines all indicator, jumper, and strap locations, and describes their function and proper position.
- Section 6** Provides installation procedures for the emulator boards and prototype control probe. This section also provides software installation procedures for the development system.
- Section 7** Describes procedures used to verify the operational and functional performance of the 80186/80188 Emulator.

Definition of Terms

This manual uses, the following terms:

- References to a development system apply to an 8540 Intergration Unit (UI) or an 8550 Microcomputer Development Lab (MDL). The 8550 MDL consists of the 8301 Microprocessor Development Unit and the 8501 Data Management Unit.
- References to emulator boards apply to the 80186/80188 Emulator boards (Board I, Board II, and Board III).
- References to probe or prototype control probe apply to the 80186 Prototype Control Probe or 80188 Prototype Control Probe.
- References to probe plug apply to the 80816/80188 Prototype Control Probe Plug. The probe plug is a 68-pin chip carrier plug that inserts into the prototype's microprocessor socket.
- References to emulator apply to the 80186/80188 Emulator boards with either the 80186 Prototype Control Probe or the 80188 Prototype Control Probe.

Notational Conventions

The following notational conventions are used throughout this manual.

System Commands

This example illustrates conventions for entering development system commands:

```
$ a1 0 3fff -s
  4 BLOCK(S) ALLOCATED    12 BLOCK(S) FREE
```

- The \$ sign represents the development system's prompt.
- All command lines must be entered from your terminal exactly as shown. Be sure to enter any spaces shown.
- You must press the RETURN key at the end of each command line.
- The system response to a command is displayed on the next line. System responses are not preceded by the prompt sign.

Hexadecimal

All addresses are in hexadecimal notation unless specified otherwise. Where it is necessary, hexadecimal numbers are followed by an H; for example 43ACH.

Signal Lines

Signal line names followed by an L in parentheses (L) are active in the low state. Line names followed by an H in parentheses (H) are active in the high state:

RUN(L)
CS16(H)
IO(H)/M(L)

DOCUMENTATION OVERVIEW

You should be familiar with your Tektronix development system and with the manuals described in the following paragraphs.

Users Manuals

Users manuals describe procedures for operating the development system and its peripheral devices. Users manuals are provided as a standard accessory to the product.

You should be familiar with these manuals about the development systems and their capabilities:

- *8550 Microcomputer Development Lab System Users Manual*
- *8540 Integration Unit System Users Manual*
- *8560 Series Multi-User Software Development Unit TNIX System Users Manual*

Instruction Manuals

Users and installation instruction manuals contain both emulator-specific user information and installation information. These manuals tell how to install the emulator hardware and software, and how to verify its operation. Instruction manuals are a standard accessory. This instruction manual is available for the 80186/80188 Emulator:

- *80186/80188 Emulator Users and Installation Manual*

Installation Manuals

Installation manuals or guides tell how to unpack, install, and verify the operation of the equipment. Installation manuals are a standard accessory that may be provided as supplements to existing publications or as separate manuals.

These installation manuals are available for the 80186/80188 Emulator's host development system:

- *8550 Microcomputer Development Lab Installation Guide*
- *8540 Integration Unit Installation Guide*

Service Manuals

Service manuals provide information necessary for system testing, isolation of hardware problems, and repair of system components. Service manuals are optional accessories and may be purchased from Tektronix.

These service manuals support the 80186/80188 Emulator, its 80186 and 80188 Prototype Control Probes, and their host development system:

- *8301 Microprocessor Development Unit Service Manual*
- *8540 Integration Unit Service Manual*
- *80186/80188 Emulator Service Manual*

REVISION INFORMATION

As this manual is revised and reprinted, revision history is included in the manual's text and diagrams. Revised pages have the word REV and a date (REV AUG 1984) at their bottom inside corner. New pages (containing old, new, or revised information) added to a section have the word ADD along with the revision date (ADD AUG 1984).

You may receive change pages or notices with this manual reflecting post-publication changes. This information may be located behind the yellow tab at the rear of the manual or may be provided separately. You will receive directions for entering the pages or replacing existing pages in the manual.

CONTENTS

	Page
Operators Safety Summary	xv
Servicing Safety Summary	xvii
 Section 1 GENERAL INFORMATION	
Introduction	1-1
Modes Of Operation	1-1
Emulator Overview	1-2
Microprocessors Supported	1-3
Emulator Hardware Configuration	1-3
Clock Rate	1-3
Emulator Boards	1-3
Prototype Control Probe	1-4
Prototype Control Probe LEDs	1-4
 Section 2 EMULATOR-SPECIFICS OPERATOR INFORMATION	
Introduction	2-1
Symbolic Debug	2-1
Byte/Word Parameter	2-1
Register Set	2-1
Status Word Register	2-3
Internal Peripheral Control Block Interface	2-6
DMA Channels	2-8
DMA Operation	2-9
DMA Control Word Registers	2-9
DMA Transfer Count Registers	2-12
DMA Destination and Source Pointer Registers	2-12
Chip-Selects and Ready Generation	2-12
Chip-Select Outputs	2-13
Upper Memory Chip-Select	2-14
Lower Memory Chip-Select	2-15
Mid-Range Memory Chip-Selects	2-16
Peripheral Chip-Selects	2-18
Ready Mode Generation	2-19
Timers	2-20
Timer Operation	2-21
Timer Mode/Control Word Register	2-22
Count Registers	2-24
Maximum Count Registers	2-25
Timers and Reset	2-25

Section 2 EMULATOR-SPECIFICS OPERATOR INFORMATION (Cont.)	Page
Interrupt Controller	2-25
NON-RMX Mode Operation	2-26
NON-RMX Mode Interrupt Controller Registers	2-26
In-Service Register	2-27
Interrupt Request Register	2-27
Mask Register	2-28
Priority Mask Register	2-28
Interrupt Status Register	2-28
Timer and DMA Control Registers	2-29
External Interrupt Control Registers	2-29
EOI Register	2-31
Poll and Poll Status Registers	2-32
RMX 8086 Compatibility Mode	2-33
RMX Mode Interrupt Controller Registers	2-33
EOI Register	2-34
In-Service Register	2-35
Interrupt Request Register	2-35
Mask Register	2-35
Timer/DMA Control Registers	2-35
Interrupt Vector Register	2-36
Priority Level Mask Register	2-37
Interrupt Controller and Reset	2-37
Peripheral Control Block Register Summary	2-38
Operating System Commands	2-39
Command Syntax	2-40
Notation Conventions	2-40
Command Line	2-40
AL—Allocates physical memory	2-41
BK—Sets or displays breakpoints	2-42
Memory Manipulation Commands D, F, LO, MOV, P, and SAV	2-44
DEAL—Deallocates physical memory	2-44
DI—Disassembles object code	2-46
DS—Displays emulator registers status	2-47
NON-RMX Mode	2-49
RMX Mode	2-50
G—Begins program execution	2-51
LPCB—Locates peripheral control block	2-52
MAP—Sets or displays memory map	2-53
MEM—Specifies available user memory	2-54
MEMSP—Defines memory space	2-54
NOMEM—Specifies unavailable user memory	2-55
RD—Reads from emulator's I/O port	2-56
RESET—Reinitializes emulator	2-57
S—Assign value to register or symbol	2-58
SEL—Selects the emulator	2-58
SPCB—Sets peripheral control block	2-60
TRA—Controls display of executed instructions	2-62
WRT—Writes to emulator's I/O port	2-69
X—Loads and executes program	2-70

Section 2 EMULATOR-SPECIFICS OPERATOR INFORMATION (Cont.)	Page
Error Messages	2-70
Real-Time Prototype Analyzer (RTPA)	2-70
Trigger Trace Analyzer (TTA) Commands	2-71
TTA Bus Operation Designators—BUS and EVE	2-71
Set Consecutive Events—CONS	2-71
Display Contents of TTA's Acquisition Memory—DISP	2-72
Special Considerations	2-74
The HALT Instruction	2-74
Emulator Fault Error	2-74
Interrupt Status Register Constraints	2-75
PROM Programmer Considerations	2-75
Emulating the Execution Bus	2-75
Prototype Interrupts During Trace All Mode	2-75
DMA Activity During Trace All Mode	2-76
Short Instructions During Trace All Mode	2-76
TTA Display Changes During DMA Activity	2-76
TTA Constraints	2-76
Locating the Stack Pointer	2-77
Segment Registers	2-77
Register Symbols—CSX, DSX, SSX and ESX	2-77
Service Calls	2-78
SVCs in Modes 1 and 2	2-79
SRB Format	2-79
SVC Demonstration	2-79
 Section 3 EMULATOR DEMONSTRATION RUN	
Introduction	3-1
Demonstration Program	3-1
Examine the Demonstration Program	3-1
Set the Table Pointer	3-2
Set the Pass Counter	3-3
Clear the Accumulator	3-3
Add a Byte from the Table	3-3
Point to the Next Byte	3-3
Decrement CX and Loop Until CX = 0	3-3
Call Exit SVC	3-3
Development System Configurations	3-4
Assemble and Load the Demonstration Program	3-4
Case 1: Assemble and Load on the 8550	3-4
Start Up and Log In	3-6
Copy the Demonstration Run Program from the Installation Disk	3-6
Examine the Demonstration Program	3-7
Assemble the Source Code	3-8
Link the Object Code	3-8
Select the 80186/80188 Emulator	3-9

Section 3 EMULATOR DEMONSTRATION RUN (Cont.)	Page
Load the Program into Memory	3-9
Allocate Memory	3-9
Zero Out Memory	3-9
Check that Memory is Filled with Zeros	3-10
Load the Object Code into Memory	3-10
Load the Program Symbols	3-10
Case 2: Assemble on the 8560; Download to the 8540 or 8550	3-11
Start Up and Log In	3-11
Create the Demonstration Program	3-11
Enter the Text	3-12
Check for Errors	3-12
Assemble the Source Code	3-14
Link the Object Code	3-14
Download the Program to the 8540	3-15
Allocate Memory	3-15
Zero Out Memory	3-15
Check that Memory is Filled with Zeros	3-15
Download the Object Code	3-16
Download the Program Symbols	3-16
Case 3: Download from Your Host to the 8540 or 8550	3-17
Create the Extended Tekhex Load Module	3-18
Start Up and Log In	3-18
Allocate Memory	3-18
Zero Out Memory	3-18
Check that Memory is Filled with Zeros	3-20
Download the Load Module	3-20
Case 4: Patch the Program into Memory	3-21
Start Up and Log In	3-21
Allocate Memory	3-21
Zero Out Memory	3-21
Check that Memory is Filled with Zeros	3-22
Patch the Object Code into Memory	3-22
Put Symbols into the Symbol Table	3-23
Run the Demonstration Program	3-23
Check Memory Contents Again	3-24
Turn On the Symbolic Display	3-24
Disassemble the Object Code	3-25
Put Values into the Table in Memory	3-26
Check the Contents of the Table	3-27
Check the Code Segment Register	3-27
Start Program Execution	3-28
Monitor Program Execution	3-28
Trace All Instructions	3-28
Trace to the Line Printer	3-31
Trace Jump Instructions Only	3-31
Set a Breakpoint after a Specific Instruction	3-33
Set Breakpoints to Stop Program Execution	3-33
Set New Values in the Pass Counter and Table Pointer	3-34
Resume Program Execution	3-35

Section 3 EMULATOR DEMONSTRATION RUN (Cont.)	Page
Summary of 80186 Emulator Demonstration Run	3-36
Delete the Demonstration Run Files	3-36
Delete 8550 Files	3-37
Delete 8560 Files	3-37
Turn Off Your System	3-37

WARNING

The following servicing instructions are for use by qualified personnel only. To avoid personal injury do not perform any servicing other than that contained in the operating instructions unless you are qualified to do so. Refer to the operators safety summary and service safety summary prior to performing any service.

Section 4 TECHNICAL INFORMATION

Introduction	4-1
Emulator Timing	4-1
Power Supply Calibration	4-9
Equipment Required	4-9
Procedure	4-10
Specifications	4-12
Electrical Characteristics	4-12
Environmental Characteristics	4-13
Physical Characteristics	4-13

Section 5 JUMPERS

Introduction	5-1
Emulator Boards Indicator, Jumpers, and Strap	5-1
Board I Jumper and Strap	5-1
P1091	5-1
W4037	5-2
Board II Indicator and Jumpers	5-2
DS3077	5-2
P1011	5-3
P2089	5-3
P6102	5-3
P7105	5-3
Prototype Control Probe Indicators and Jumpers	5-4
Control Board Jumpers	5-5
P4023	5-5
Buffer Board Jumpers	5-6
P4023 and P6023	5-6
P5023	5-6
P7023	5-7
P8027	5-7
P9011	5-7
P9013	5-8
P9015	5-9

Section 5 JUMPERS (Cont.)	Page
Power Supply Board Indicators and Jumper	5-10
DS1021	5-10
P5061	5-10
CPU Board Jumpers	5-11
P2067	5-11
P2068	5-12
Dummy Address Jumpers	5-12
P6042	5-13
P6043	5-13
Development System Jumpers and Straps	5-14
32K Program Memory Board Jumper Adjustments	5-14
64K/128K Program Memory Board Jumper Adjustments	5-14
80186/80188 and Development System Jumper Summary	5-15

Section 6 INSTALLATION

Introduction	6-1
Hardware Installation	6-1
Installing the Emulator Boards and Prototype Control Probe	6-2
Connecting to the Prototype	6-8
Probe Plug Insertion Procedure	6-8
Prototype Control Probe Disassembly Procedure	6-10
Interface Assembly Disassembly Procedure	6-10
Top Cover Disassembly Procedure	6-11
Bottom Cover Disassembly Procedure	6-11
Connecting to the Trigger Trace Analyzer (Optional)	6-12
Grounding	6-13
Software Installation	6-13
Setting up the 8560	6-13
Setting Your Shell Variable	6-13
Installing the 8540 Firmware	6-14
Installing the 8550 Software	6-17
Start Up and Set the Date	6-17
Installation Procedure	6-18
Installing the Diagnostic Software	6-19
Using Your 8086/8088 Assembler on the 8550	6-19

Section 7 VERIFICATION PROCEDURES

Introduction	7-1
Operational Test Procedure	7-1
Equipment Required	7-1
Equipment Setup	7-2
Functional Test Procedure	7-2
No HELLO Display	7-3
Processor Test Procedure	7-3
Emulator Timing Verification	7-4
Measurement Considerations	7-4
Equipment Required	7-5
Controlling the Signal Lines Under Test	7-5

Section 7 VERIFICATION PROCEDURES (Cont.)	Page
System Functional Verification	7-5
8550 Microcomputer Development Lab Verification	7-6
Procedure	7-6
8540 Integration Unit Verification	7-7
Procedure	7-7

ILLUSTRATIONS

Figure No.		Page
1-1	Emulation modes 0, 1, and 2	1-2
1-2	80186/80188 Emulator interconnections	1-5
2-1	80816/80188 register set	2-2
2-2	Status Word (Flags) Register	2-3
2-3	Peripheral Control Block's internal register map	2-7
2-4	Relocation Register content	2-8
2-5	DMA Control Word Register	2-10
2-6	DMA Destination Pointer and Source Pointer register format	2-12
2-7	UMCS Register format	2-14
2-8	LMCS Register format	2-15
2-9	MPCS Register format	2-16
2-10	MMCS Register format	2-17
2-11	PACS Register format	2-18
2-12	Timer Mode/Control Word Register format	2-22
2-13	In-Service, Interrupt Request, and Mask register formats	2-27
2-14	Priority Mask Register format	2-28
2-15	Interrupt Status Register format	2-29
2-16	Timer/DMA Control register formats	2-29
2-17	INT0 and INT1 Control register formats	2-30
2-18	INT2 and INT3 Control register formats	2-30
2-19	EOI Register format	2-32
2-20	Poll and Poll Status register formats	2-32
2-21	EOI Register format	2-34
2-22	In-Service, Interrupt Request, and Mask register formats	2-35
2-23	Timer/DMA Control register formats	2-36
2-24	Interrupt Vector Register format	2-36
2-25	Priority Level Mask Register format	2-37
2-26	Sample syntax block	2-40
2-27	80186/80188 SVC demonstration program listing	2-81
3-1	80186 demonstration run program	3-2
3-2	Development system configurations	3-5
3-3	Host computer commands for preparing the demonstration program	3-17
3-4	80186 demonstration run program: extended Tekhex format	3-19
4-1	80186 write cycle timing diagram	4-4
4-2	80186 read cycle timing diagram	4-5
4-3	Clock timing diagram	4-6

ILLUSTRATIONS (Cont.)

Figure No.		Page
4-4	HOLD and HOLDA timing diagram	4-7
4-5	80186 timer timing diagram	4-8
4-6	Probe Power Supply Board	4-11
5-1	Board I jumper and strap locations	5-2
5-2	Board II jumper and indicator locations	5-4
5-3	Control Board jumper locations	5-5
5-4	Buffer Board jumper and fuse locations	5-9
5-5	Power Supply Board jumper and indicator locations	5-11
5-6	CPU Board jumper locations	5-13
6-1	Removal/installation of the top cover	6-2
6-2	80186/80188 Emulator board interconnections	6-3
6-3	Recommended board arrangement for the 8301	6-4
6-4	Recommended board arrangement for the 8540	6-4
6-5	Strain relief plate installation/removal	6-5
6-6	Ribbon cable installation	6-6
6-7	Emulator boards with prototype control probe	6-7
6-8	Pin identification and proper plug insertion	6-9
6-9	80186/80188 Emulator Board I connections to the TTA	6-12
6-10	System ROM Board socket locations	6-14
6-11	Type-2764 and type-27128 ROM strapping	6-16

TABLES

Table No.		Page
2-1	Status Word (Flags) Bit Functions	2-4
2-2	Registers and Flags	2-5
2-3	DMA Register Offset Addresses and Symbol Names	2-9
2-4	Chip-Select Register Offset Addresses and Symbol Names	2-13
2-5	UMCS Programming Values	2-14
2-6	LMCS Programming Values	2-15
2-7	MPCS Programming Values	2-16
2-8	Peripheral Chip-Select Address Ranges	2-18
2-9	Bits MS and EX Programming Values	2-19
2-10	Ready Mode Programming Bits	2-20
2-11	Timer Register Offset Addresses and Symbol Names	2-22
2-12	Interrupt Controller Registers (NON-RMX Mode)	2-26
2-13	80186 Interrupt Vectors	2-31
2-14	RMX Mode Internal Interrupt Source Priority Level	2-33
2-15	Interrupt Controller Registers (RMX Mode)	2-34
2-16	Symbol Names for DMA, Chip-Select, and Timer Registers	2-38
2-17	Symbol Names for Interrupt Control Registers	2-39
2-18	80186/80188 TTA Bus Operation Designators	2-71
2-19	80186/80188 Service Calls	2-80
3-1	Basic 8560 Editing Commands	3-13
4-1	80186 Emulator Timing Differences	4-2
4-2	Electrical Characteristics	4-12
4-3	Environmental Characteristics	4-13
4-4	Emulator Physical Characteristics	4-13
5-1	Ready Fault/Time-out Jumpers	5-8
5-2	Dummy Address Jumpers	5-12
5-3	80186/80188 Jumper Default Position Summary	5-15
5-4	System Jumper Default Position Summary	5-16
7-1	Stand-Alone Processor Tests	7-4

OPERATORS SAFETY SUMMARY

The general safety information in this summary is for operating and servicing personnel. Specific warnings and cautions will appear throughout the manual.

IN THIS MANUAL

CAUTION statements identify conditions or practices that could result in damage to the equipment or other property. WARNING statements identify conditions or practices that could result in personal injury or loss of life.

MARKED ON EQUIPMENT

CAUTION indicates a personal injury hazard not immediately accessible as you read the marking, or a hazard to property including the equipment. DANGER indicates a personal injury hazard immediately accessible as you read the marking.

SYMBOLS MARKED ON EQUIPMENT



DANGER high voltage



Protective ground (earth) terminal



Attention - refer to manual



Susceptible to damage from static charge

SAFETY PRECAUTIONS

Follow these safety precautions while using this product.

Ground the Product

The product is grounded through the grounding conductors in the interconnecting cables. To avoid electrical shock, plug the supporting system's power cord into a properly wired and grounded socket. You must have protective ground connection to operate this product safely.

Use the Proper Fuse

To avoid fire hazard, use only the fuse specified in the parts list for your product. Be sure the fuse is identical in type, voltage rating, and current rating.

Only qualified service personnel should replace fuses.

Do Not Operate in Explosive Atmospheres

To avoid explosion, do not operate this product in an atmosphere of explosive gases unless such operation has been specifically certified.

Do Not Remove Covers or Panels

To avoid personal injury, do not remove the product covers or panels. Do not operate the product without the covers and panel properly installed.

SERVICING SAFETY SUMMARY

SERVICING INFORMATION IS FOR QUALIFIED SERVICE PERSONNEL ONLY.

Follow these safety precautions while servicing this product. Refer to the Operators Safety Summary for general safety information.

DO NOT SERVICE ALONE

Do not perform internal service or adjustment on this product unless another person able to give first aid and resuscitation is present.

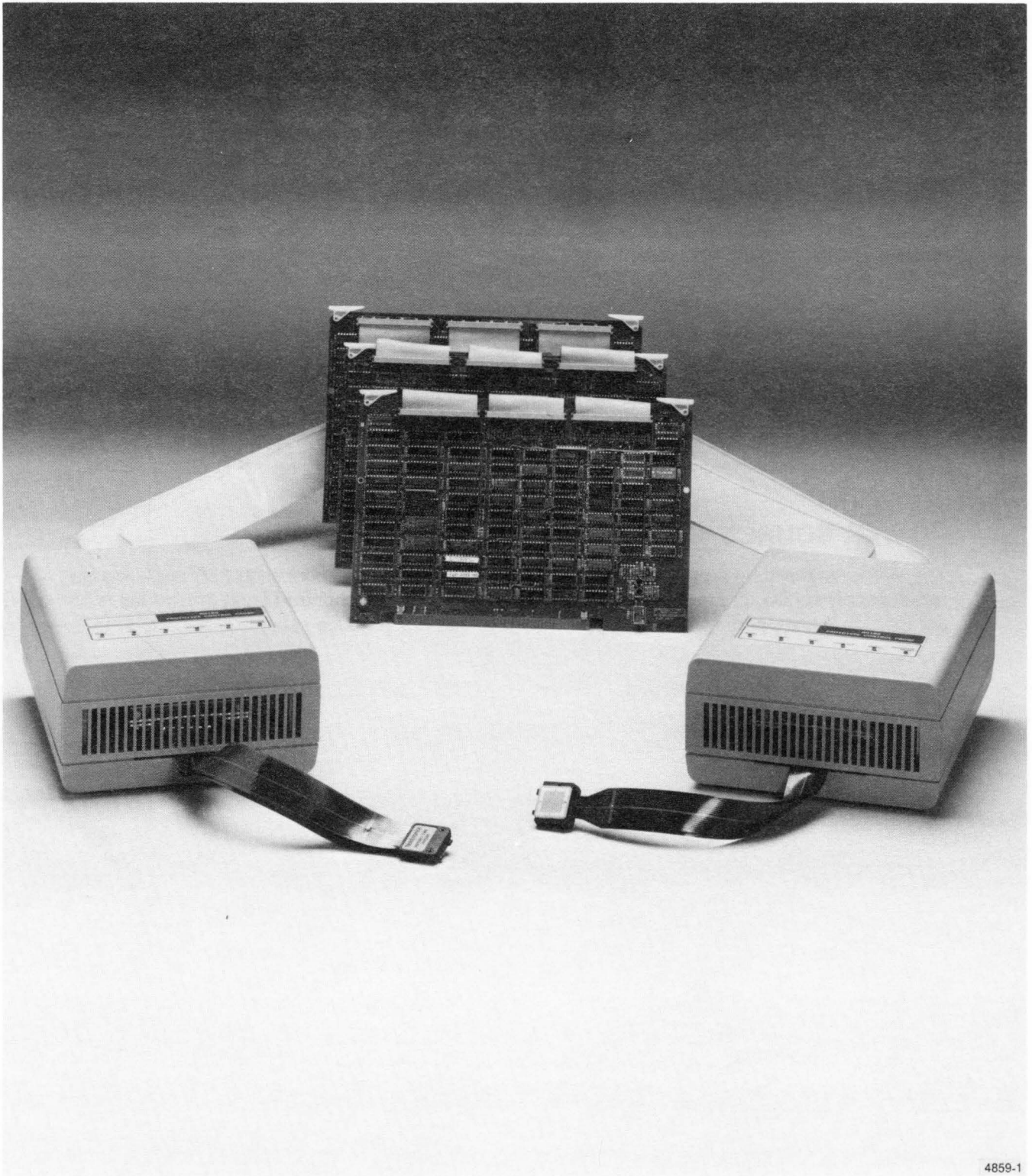
USE CARE WHEN SERVICING WITH POWER ON

To avoid personal injury from dangerous voltages, remove jewelry such as rings, watches, and other metallic objects before servicing. Do not touch the product's exposed connections and components while the power is on.

Disconnect the power before removing protective panels, soldering, or replacing components.

POWER SOURCE

The supporting system's primary power source should not apply more than 250 volts rms between supply conductors, or between supply conductors and ground. You must ground the product through the supporting system's power cord to use this produce safely.



4859-1

80186/80188 Emulator

Section 1

GENERAL INFORMATION

INTRODUCTION

The 80186/80188 Emulator is an option designed to function as part of the 8500 Series development systems. The 80186/80188 Emulator provides in-circuit emulation for an 80186 or an 80188 microprocessor.

This instruction manual is divided into two parts. The first part contains specific user and operating information for the 80186/80188 Emulator. The second part contains procedures for emulator hardware and control software installation. In addition, the second part contains procedures for performing product verification for the emulator in either an 8540 or an 8550 development system. Installation and verification procedures are contained in Sections 6 and 7 of this manual. For information about the development system, refer to your System Users Manual. Servicing information is contained in the *80186/80188 Emulator Service Manual*.

This section explains the emulation modes and gives an overview of the emulator.

NOTE

The 80186 or 80188 microprocessor contained within the prototype control probe is static-sensitive. A static discharge can interrupt the normal operation of the emulator. Avoid touching the prototype control probe cables and 68-pin probe plug, while the emulation system is operating.

MODES OF OPERATION

The 80186/80188 Emulator operates in one of three emulation modes, and provides in-circuit emulation for an 80186 or an 80188 microprocessor. Figure 1-1 illustrates the three emulation modes described here.

Mode 0 System mode. Use mode 0 to develop software for a prototype circuit based on an 80186 or 80188 microprocessor. In mode 0, the development system acts as a stand-alone, 80186- or 80188-based microcomputer. The prototype software is stored in your development system's program memory, and the 80186/80188 Emulator provides the I/O facilities and clock.

Mode 1 Partial emulation mode. Use mode 1 to develop some of the prototype circuit's hardware functions. In mode 1, the prototype control probe is connected between the 80186/80188 Emulator boards and the microprocessor socket in the prototype circuit. Mode 1 exercises the prototype's memory, I/O, and clock while the development system retains full control over the prototype system.

In mode 1, the memory mapping feature allows portions of the prototype software to be stored in prototype memory. The rest of the prototype software remains in the development system's program memory.

Mode 2 Full emulation mode. Use mode 2 in the final software/hardware integration phases of prototype system design. You may also use this mode when you are troubleshooting hardware with error-free software. The only difference between mode 1 and mode 2 is that in mode 2, the entire prototype program is stored in the prototype's memory, and the development system's program memory cannot be accessed.

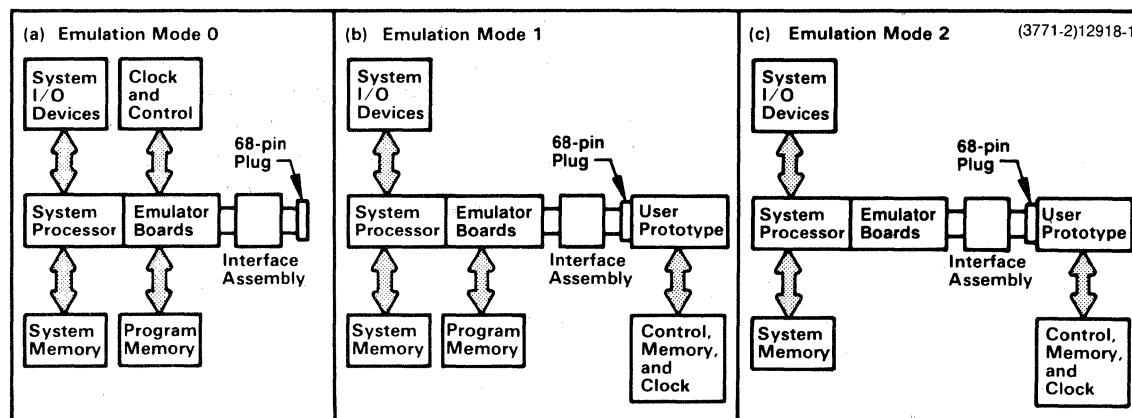


Figure 1-1. Emulation modes 0, 1, and 2.

EMULATOR OVERVIEW

The 80186/80188 Emulator consists of three emulator boards (Board I, Board II, and Board III) and either an 80186 Prototype Control Probe or an 80188 Prototype Control Probe. The three emulator boards plug into the development system's mainframe, and the prototype control probe attaches to Emulator Board II.

The 80186/80188 Emulator serves two purposes in the microcomputer development system. First, the emulator can run a program written specifically for the target microprocessor. With the help of other boards in the system, the emulator can check the program for run-time errors or program-logic errors. Second, with the prototype control probe connected between the emulator boards and the prototype circuit, the prototype circuit can be debugged and stepped through the final stages to design completion.

The 80186/80188 Emulator emulates the operation of a target microprocessor device for the final version of a prototype system. The emulator responds to software the same way the target microprocessor responds, but the emulator also allows software debugging.

Microprocessors Supported

The 80186 Emulator emulates the Intel 80186 microprocessor.

The 80188 Emulator emulates the Intel 80188 microprocessor.

The 80186 and 80188 microprocessors differ only in their data bus width. The 80186 uses a 16-bit or an 8-bit data bus width, and the 80188 uses an 8-bit data bus width only.

Emulator Hardware Configuration

In emulation mode 0, the prototype control probe is connected to the emulator boards. Your program runs in 8550 or 8540 program memory. In emulation modes 1 and 2, the prototype control probe must be connected to both the emulator boards and your prototype. For instructions on installing your emulator and probe, refer to Section 6 of this manual.

Clock Rate

The input frequency to the emulator's clock (used in emulation mode 0) is 8 MHz or 16 MHz. The maximum input frequency from the prototype's clock (used in emulation modes 1 and 2) is 16 MHz.

Emulator Boards

The three 80186/80188 Emulator boards plug into the Main Interconnect Board of the development system's mainframe. The 80186/80188 Emulator boards are located in the development system as follows:

- | | |
|-----------|--|
| Board I | This circuit board plugs into the first of three adjacent slots (J14) on the "program" side of the development system's Main Interconnect Board. |
| Board II | This circuit board plugs into the second of three adjacent slots (J15) on the program side of the development system's Main Interconnect Board. |
| Board III | This circuit board plugs into the third of three adjacent slots (J16) on the program side of the development system's Main Interconnect Board. |

Prototype Control Probe

The prototype control probe allows an emulator to functionally replace a prototype system's microprocessor. The probe plug is the interface between the emulator boards and the prototype circuit's microprocessor socket.

The prototype control probe consists of:

1. Two 6-foot, 40-conductor ribbon cables that attach to Emulator Board II.
2. An interface assembly, which contains the Buffer Board, Control Board, CPU Board, and Power Supply Board.
3. A 68-pin chip carrier plug that inserts into the prototype circuit board's microprocessor socket. The plug is connected to the interface assembly by four 20-conductor, 1-foot ribbon cables.

Figure 1-2 shows the 80186/80188 Emulator interconnections.

Prototype Control Probe LEDs

Six indicator LEDs are visible on the prototype control probe's interface assembly. When lit, these LEDs indicate the following conditions:

RESET	The prototype circuit has asserted the RESET line and caused the processor to stop current activity.
TEST	The prototype circuit has asserted the TEST line. (This LED is always lit in mode 0.)
WAIT	The emulating processor is in a wait state.
HOLD ACK	The emulating processor is acknowledging a bus request.
SELECT	The emulator has been selected.
PROTOTYPE POWER	The prototype circuit's +5 Vdc supply is present.

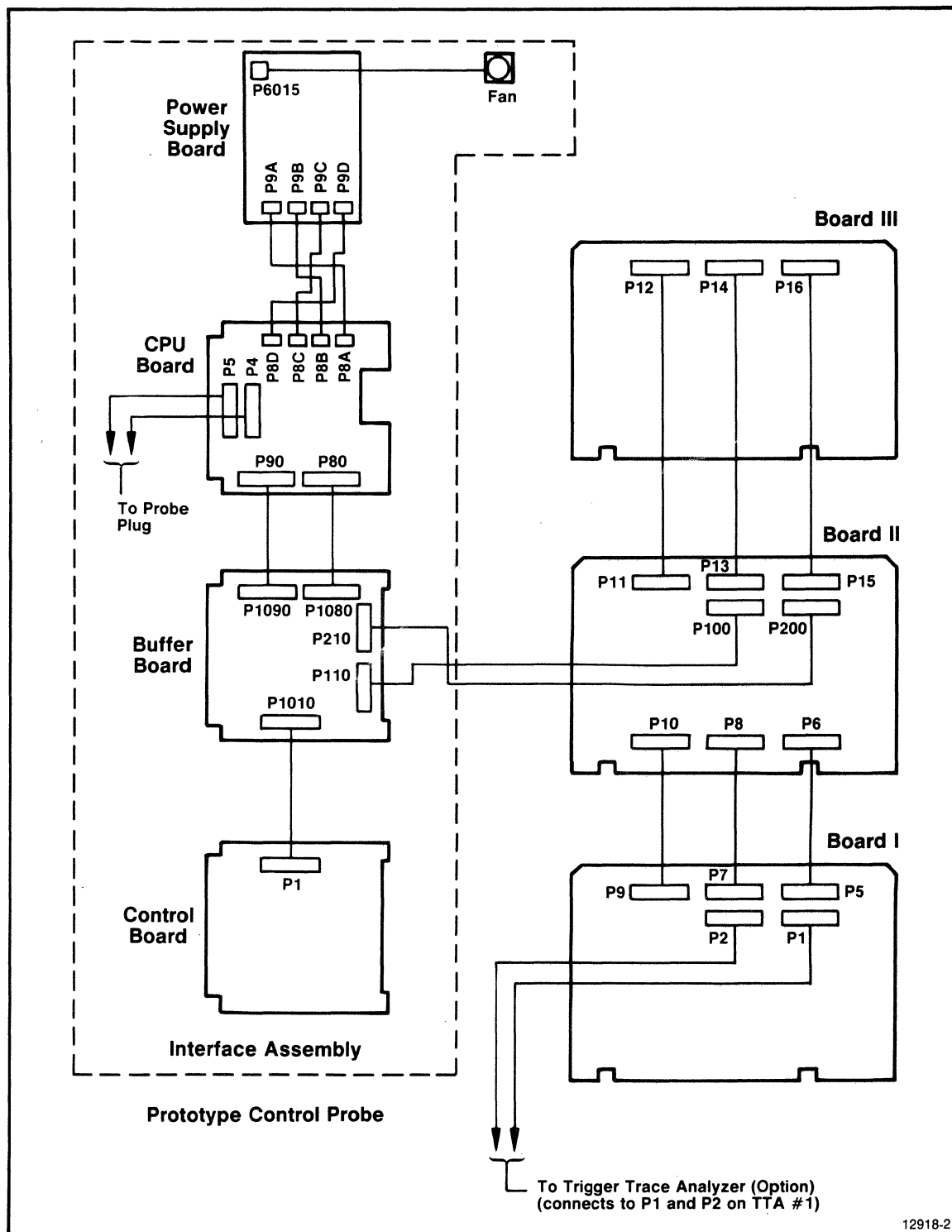


Figure 1-2. 80186/80188 Emulator interconnections.

Section 2

EMULATOR-SPECIFICS OPERATOR INFORMATION

INTRODUCTION

This section explains the features of the 8550 and 8540 development systems that are unique to the 80186 Emulator and the 80188 Emulator. Throughout this section, references to your System Users Manual apply to your *8550 System Users Manual* or *8540 System Users Manual*.

Symbolic Debug

The 80186/80188 Emulator supports the use of symbolic debug. Many displays in this section include symbolic debug information. Refer to the section discussing emulation in your System Users Manual for information about symbolic debug.

Byte/Word Parameter

Several commands in this section offer you the choice of operating in memory on a byte-oriented or word-oriented basis. For those commands that permit a byte/word parameter, the byte parameter uses a **-b** modifier and the word parameter uses a **-w** modifier. The default value is **-b**, for the 80186/80188 Emulator.

REGISTER SET

The basic architecture for the 80186 or 80188 microprocessor device contains fourteen 16-bit special registers that are divided into three functional categories. Figure 2-1 shows the register name, category, and special function of each register.

General	Eight 16-bit general purpose registers store arithmetic and logical operands. Four of these registers (AX, BX, CX, and DX) are configured as 16-bit registers, but each can be divided into two separate 8-bit registers.
---------	---

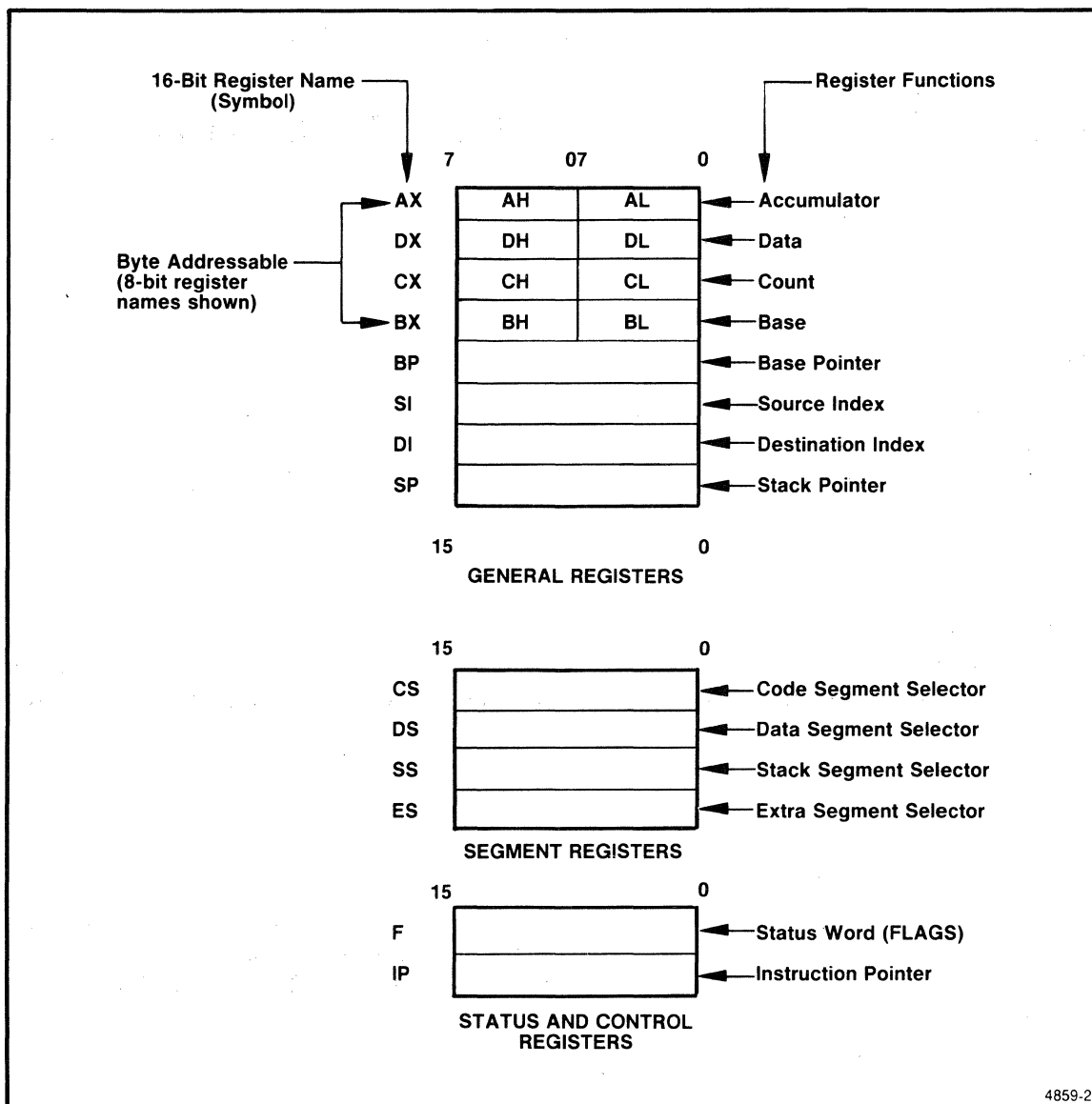


Figure 2-1. 80816/80188 register set.

Segment Four 16-bit segment registers address four segments of memory with each segment containing 64K bytes of addressable memory. The memory segments are addressed by adding a 16-bit offset to the 16-bit address in one of the segment registers. This permits a physical address size up to 1M byte.

Status and Control Two 16-bit special purpose registers provide status and control. The Status Word (Flags) Register contains the status and control flag bits as shown in Figure 2-2. The Instruction Pointer Register contains the offset address of the next sequential instruction to be executed.

Status Word Register

Figure 2-2 shows the format of the Status Word (Flags) Register. Table 2-1 describes the Status Word bit functions.

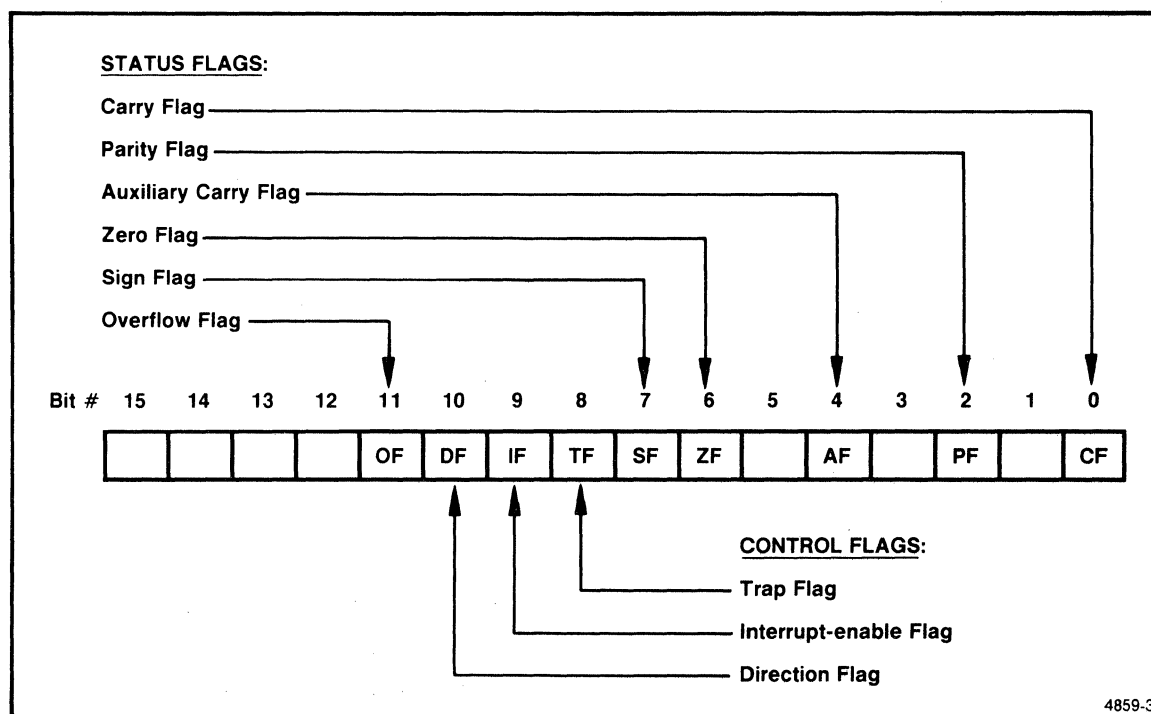


Figure 2-2. Status Word (Flags) Register.

Table 2-1
Status Word (Flags) Bit Functions

Status Bit	Symbol	Function
0	CF	Carry Flag. This bit is set to 1, when a high-order bit carry or borrow occurs. This bit is cleared (0) at other times.
2	PF	Parity Flag. This bit is set to 1, when the low-order result (8 bits) contains an even number of 1-bits. The bit is cleared (0) at other times.
4	AF	Auxiliary Carry Flag. This bit is set to 1, when a carry from or borrow to the low-order four bits of AL occurs. This bit is cleared (0) at other times.
6	ZF	Zero Flag. This bit is set to 1, when the result is 0. The bit is cleared (0) at other times.
7	SF	Sign Flag. This bit is set equal to the result's high-order bit; set to 0 if positive and to 1 if negative.
8	TF	Trap (Single Step) Flag. When this flag is set to 1, a single step interrupt occurs after the next instruction executes. Cleared by the single step interrupt.
9	IF	Interrupt-enable Flag. When this flag is set to 1, maskable interrupts cause the processor to transfer control to a specified interrupt vector location.
10	DF	Direction Flag. When set to 1, this flag causes string instructions to automatically decrement the appropriate Index Register. When this flag is clear (0), the Index Register is automatically incremented.
11	OF	Overflow Flag. Set to 1 if the signed result cannot be expressed within the number of bits in the destination operand. Cleared (0) at other times.

The register and flag symbols shown in Figures 2-1 and 2-2, and in Table 2-1 are used by DOS/50 and OS/40 to designate a specific register or flag used with the 80186/80188 Emulator. Table 2-2 alphabetically lists these register and flag symbols and provides a brief description of each symbol.

Table 2-2
Registers and Flags

Symbol	Description	Size in Bits	Value After Reset ^a	Altered by s command?
AF	Auxiliary Carry Flag ^b	1	0	yes
AH	High-order byte of Register A	8	NC	yes
AL	Low-order byte of Register A	8	NC	yes
AX	Register A	16	NC	yes
BH	High-order byte of Register B	8	NC	yes
BL	Low-order byte of Register B	8	NC	yes
BP	Base Pointer Register	16	NC	yes
BX	Register B	16	NC	yes
CF	Carry Flag ^b	1	0	yes
CH	High-order byte of Register C	8	NC	yes
CL	Low-order byte of Register C	8	NC	yes
CS	Code Segment Register	16	FFFF	yes
CX	Register C	16	NC	yes
DF	Direction Flag ^b	1	0	yes
DH	High-order byte of Register D	8	NC	yes
DI	Destination Index Register	16	NC	yes
DL	Low-order byte of Register D	8	NC	yes
DS	Data Segment Register	16	0000	yes
DX	Register D	16	NC	yes
ES	Extra Segment Register	16	0000	yes
FLAGS	Flags Register ^b	16	0000	yes
IF	Interrupt-Enable Flag ^b	1	0	yes
INTR ^c	Interrupt Request Input	1	NC	no
IP	Instruction Pointer	16	0000	no
OF	Overflow Flag ^b	1	0	yes
NMI ^c	Non-Maskable Interrupt Input	1	NC	no
PF	Parity Flag ^b	1	0	yes
SF	Sign Flag ^b	1	0	yes
SI	Source Index Register	16	NC	yes
SP	Stack Pointer Register	16	NC	yes
SS	Stack Segment Register	16	0000	yes
TEST ^c	Test control for WAIT instruction	1	NC	no
TF	Trap Flag ^b	1	0	yes
ZF	Zero Flag ^b	1	0	yes

^a NC = not changed by reset command.

^b The Flags Register is illustrated in Figure 2-2.

^c INTR, NMI, and TEST are hardware inputs to the microprocessor.

INTERNAL PERIPHERAL CONTROL BLOCK INTERFACE

In addition to the 14 internal special registers in the register set, the 80186/80188 microprocessor has an internal 256-byte control block called the Peripheral Control Block (PCB). The PCB contains a number of 16-bit registers that controls various peripherals:

- Two independent high-speed DMA channels
- Six memory chip-select outputs
- Seven peripheral device chip-select outputs
- Three internal 16-bit programmable timers
- An internal interrupt controller for both internal and external interrupts

The PCB's base address is programmed by setting the bits in the Relocation Register. The Relocation Register is the top register in the PCB, and is at an offset address FEH from the PCB's base address. Figure 2-3 shows the PCB's internal register map. The Relocation Register's bit format is shown in Figure 2-4.

The Relocation Register's lower 12 bits specify the PCB's base address. The base address of the PCB must be on an even 256-byte boundary (the lower eight bits of the base address are all 0).

The PCB can be mapped to memory or I/O space depending on the setting of bit 12. If bit 12 is high, the PCB is located in memory space. If bit 12 is low, the PCB is located in I/O space. When the PCB is mapped into I/O space, the upper four bits of the base address must be set to 0 (I/O addresses contain only 16 bits).

Besides providing relocation information for the PCB, the Relocation Register bit 14 determines the RMX or NON-RMX operating modes. If this bit is set high, the RMX mode is selected, which is an 8086 compatibility mode. When RMX mode is used, the 80186's internal Interrupt Controller is used as a slave controller to an external Master Interrupt Controller.

At reset, the Relocation Register is set to a default value of 20FF. This default value locates the PCB's base address to an address of FF00 in I/O space and sets the operating mode to NON-RMX.

The various PCB registers and register values are displayed by the **ds -p** command and set with the **spcb** command. These commands are described later in this section. Symbol names are used in the display and set commands to identify the various registers. A description of all registers within the PCB, including symbol names, offset addresses, and bit functions, is contained in the following pages. The 80186 microprocessor is used as a model, however, you can substitute the 80188 microprocessor throughout the description unless otherwise noted.

	Offset
Relocation Register	FEH
	DAH
DMA Registers Channel 1	
	D0H
	CAH
DMA Registers Channel 0	
	C0H
	A8H
Chip-Select Control Registers	
	A0H
	66H
Timer 2 Control Registers	
	60H
Timer 1 Control Registers	5EH
	58H
Timer 0 Control Registers	56H
	50H
	3EH
Interrupt Controller Registers	
	20H

4859-4

Figure 2-3. Peripheral Control Block's internal register map.

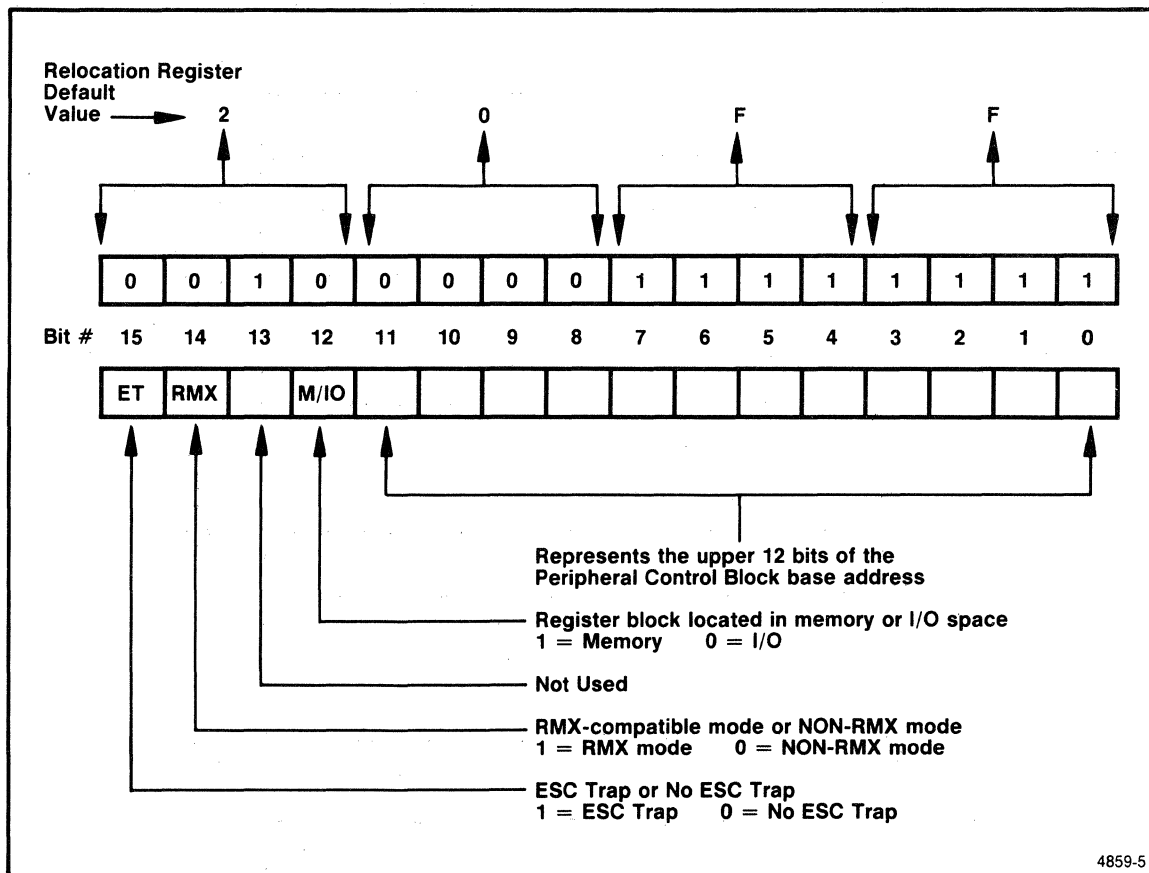


Figure 2-4. Relocation Register content.

DMA Channels

The 80186 DMA internal controller provides two independent high-speed DMA channels. The data can be transferred in bytes (8 bits) or in words (16 bits). The source address and destination address may be even or odd. Data transfers can occur between memory and I/O spaces or within the same space, from memory to memory, or I/O to I/O.

DMA Operation

Each DMA channel has six 16-bit registers that define each channel's specific operation. The six 16-bit registers make up the four following registers.

Control Word	One 16-bit register
Transfer Count	One 16-bit register
Destination Pointer	One 20-bit register (two words) configured from two 16-bit registers (12 bits of one register are not used)
Source Pointer	One 20-bit register (two words) configured from two 16-bit registers (12 bits of one register are not used)

The DMA registers' offset addresses and symbol names are shown in Table 2-3. The offset addresses are in relationship to the PCB's base address.

Table 2-3
DMA Register Offset Addresses and Symbol Names

Register Name	Offset Address	Symbol Name
Channel 0 Registers:		
Source Pointer Register	C0H and C2H	DMA0SP
Destination Pointer Register	C4H and C6H	DMA0DP
Transfer Count Register	C8H	DMA0TC
Control Word Register	CAH	DMA0CW
Channel 1 Registers:		
Source Pointer Register	D0H and D2H	DMA1SP
Destination Pointer Register	D4H and D6H	DMA1DP
Transfer Count Register	D8H	DMA1TC
Control Word Register	DAH	DMA1CW

DMA Control Word Registers

Each DMA channel contains a 16-bit Control Word Register. The control word defines the DMA channel's operating mode. Figure 2-5 shows the function of each bit in the Control Word Register. Each channel's Control Word Register may be modified or altered during any DMA activity. Changes made to these registers during an operation is reflected immediately in the current DMA transfer.

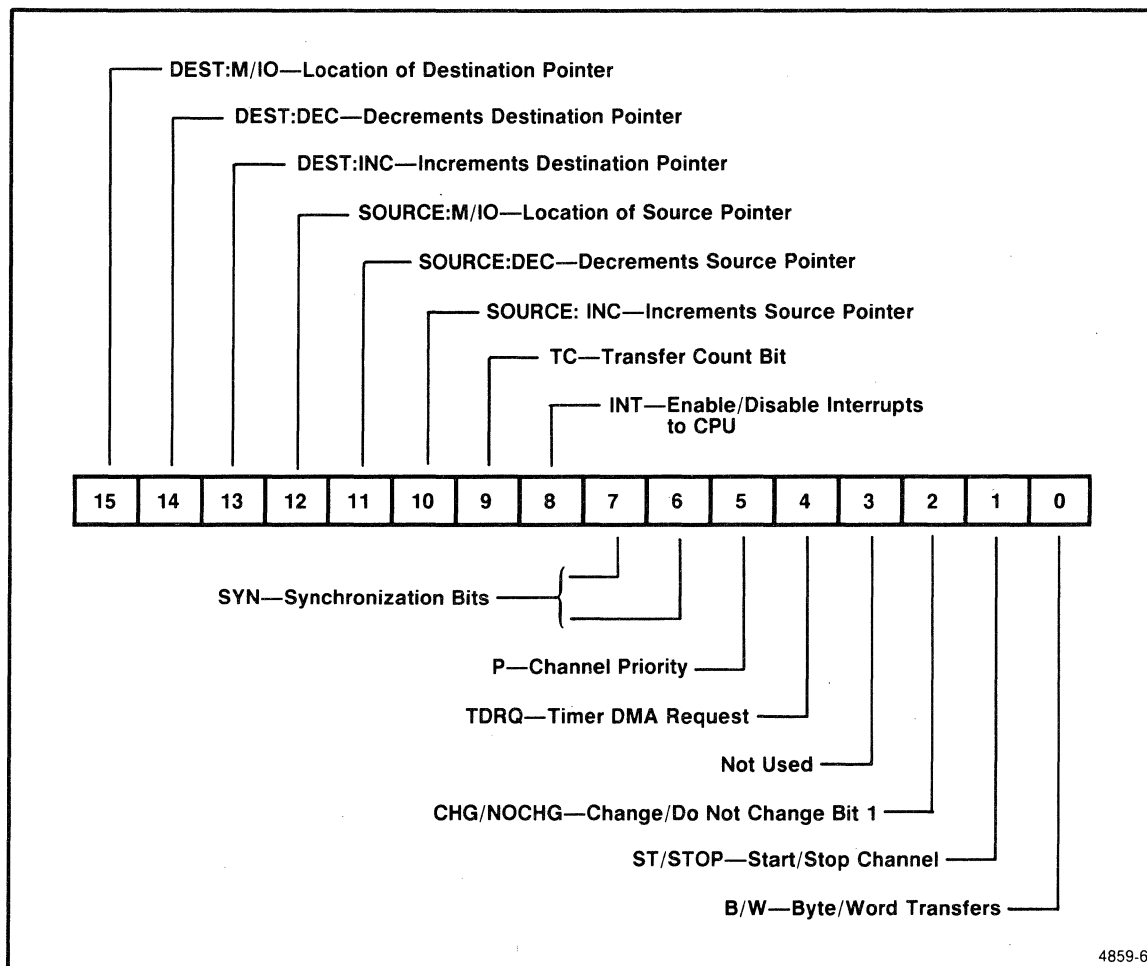


Figure 2-5. DMA Control Word Register.

The bits in the DMA Control Word Register shown in Figure 2-5 have the following functions:.

- Bit 15** **DEST:M/IO.** Designates the location of the destination pointer in either memory or I/O space.
1 = Memory space
0 = I/O space
- Bit 14** **DEST:DEC.** If set, after each DMA transfer the destination pointer is decremented by one for byte transfers or by two for word transfers. The number of bytes transferred depends on the setting of bit 0.
- Bit 13** **DEST:INC.** If set, after each DMA transfer the destination pointer is incremented by one for byte transfers or by two for word transfers. The number of bytes transferred depends on the setting of bit 0.

- Bit 12 **SOURCE:M/IO.** Designates the location of the source pointer in memory or I/O space.
 1 = Memory space
 0 = I/O space
- Bit 11 **SOURCE:DEC.** If set, after each DMA transfer the source pointer is decremented by one for byte transfers or by two for word transfers. The number of bytes transferred depends on the setting of bit 0.
- Bit 10 **SOURCE:INC.** If set, after each DMA transfer the source pointer is incremented by one for byte transfers or by two for word transfers. The number of bytes transferred depends on the setting of bit 0.
- Bit 9 **TC.** If set, the TC (Transfer Count) Register decrements after each DMA transfer cycle. DMA activity terminates when the TC Register's contents reach 0, which also resets bit 1 (ST/STOP bit). If the TC bit is cleared, the TC Register decrements after each DMA transfer cycle but DMA activity does not terminate when the TC Register's contents reach 0.
- Bit 8 **INT.** If set, enables the interrupts to the CPU when a byte count terminates.
- Bits 7
and 6 **SYN.** Synchronization bits
 00 = No synchronization
 Note: Bit 1 (ST/STOP bit) is cleared automatically when the contents of the TC Register reach 0, regardless of the state of the TC bit.
 01 = Source synchronization
 10 = Destination synchronization
 11 = Not used
- Bit 5 **P.** Sets the channel priority for each DMA channel relative to the other channel. The DMA channels alternately cycle if both are set to same priority level.
 0 = Low priority
 1 = High priority
- Bit 4 **TDRQ.** If set, this Timer DMA Request bit enables the DMA requests from Timer 2. If cleared, this bit disables the DMA requests from Timer 2.
- Bit 3 Not used.
- Bit 2 **CHG/NOCHG.** When writing to the Control Word Register, if this bit is set, bit 1 (ST/STOP bit) is also set. If this bit is cleared, when writing to the Control Word Register, bit 1 (ST/STOP) is not changed. This bit is not stored; it always reads 0.
- Bit 1 **ST/STOP.** If set, this bit starts DMA channel activity. If cleared, this bit stops DMA channel activity.
- Bit 0 **B/W.** If set, this bit increments or decrements the source pointer and destination pointer by two (word transfer) after each DMA transfer. If this bit is cleared, the pointers are incremented or decremented by one (byte transfer) after each DMA transfer.

DMA Transfer Count Registers

Each DMA channel contains a 16-bit Transfer Count (TC) Register. The register is decremented after each DMA transfer cycle, regardless of the state of the TC bit in its DMA Control Word Register. If the TC bit is set, DMA activity terminates for that channel when the TC Register reaches 0.

DMA Destination and Source Pointer Registers

Each DMA channel has a 20-bit source pointer and a 20-bit destination pointer. Each pointer occupies two full 16-bit registers in the PCB. The lower four bits of the upper register address contain the upper four bits of the 20-bit physical address. The upper 12 bits of the upper register are not used as shown in Figure 2-6.

The pointers are individually incremented or decremented after each DMA transfer cycle. If word transfers are performed, the pointers are incremented or decremented by two. If byte transfers are performed the pointers are incremented or decremented by one. Each pointer points into memory or I/O space.

Because the DMA channels can perform transfers to or from odd addresses, there are no restrictions on the source and destination values entered into the pointer registers. However, higher transfer rates are obtained when all-word transfers are performed to even addresses. This allows data to be accessed from a single memory location.

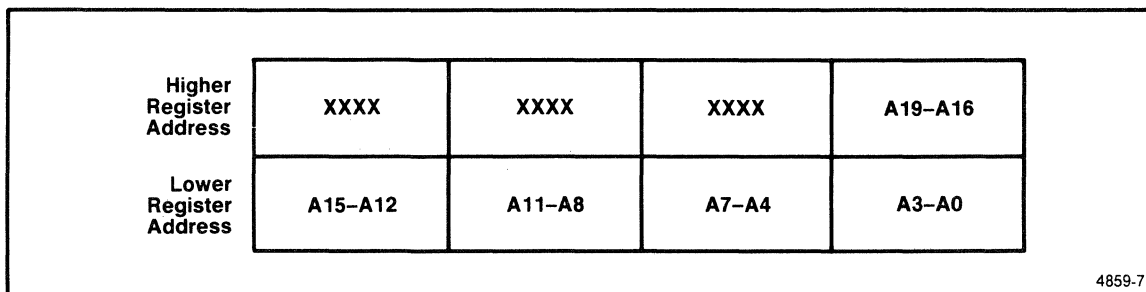


Figure 2-6. DMA Destination Pointer and Source Pointer register format.

Chip-Selects and Ready Generation

The 80186 contains logic circuitry that provides programmable chip-select generation for memory and peripherals. The 80186 can also be programmed to provide ready (wait state) generation signals. The chip-select lines are active for all memory and I/O locations in their programmed areas.

Chip-Select Outputs

The 80186 microprocessor provides six memory chip-select outputs for three address ranges and seven peripheral device chip-select outputs:

Memory Chip-Selects

Upper Memory	One chip-select output UCS(L) is generated when accessing a memory block located at the top of memory. This upper memory block is defined by the Upper Memory Chip-Select Register (UMCS Register).
Lower Memory	One chip-select output LCS(L) is generated when accessing a memory block located at the bottom of memory. This lower memory block is defined by the Lower Memory Chip-Select Register (LMCS Register).
Mid-Range Memory	One of four chip-select outputs MCS0(L)-MCS3(L) is generated when accessing a memory block located anywhere within the 1M-byte memory address space, except for the memory blocks defined by the UMCS and LMCS registers. This mid-range memory block is defined by the Mid-Range Memory Chip-Select Register (MMCS Register) and the Memory/Peripheral Chip-Select Register (MPCS Register).

Peripheral Chip-Selects

Peripheral Devices	One of seven peripheral chip-select outputs PCS0(L)-PCS6(L) is generated when accessing a peripheral chip-select memory block defined by the Peripheral Address Chip-Select Register (PACS Register) and the Memory/Peripheral Chip-Select Register (MPCS Register).
--------------------	--

Five 16-bit chip-select registers in the PCB specify the control and addresses of the preceding chip-select outputs. The chip-select registers' offset addresses and symbol names are shown in Table 2-4. The offset addresses are in relationship to the PCB's base address.

Table 2-4
Chip-Select Register Offset Addresses and Symbol Names

Register Name	Offset Address	Symbol Name
Upper Memory Chip-Select Register	A0H	UMCS
Lower Memory Chip-Select Register	A2H	LMCS
Peripheral Address Chip-Select Register	A4H	PACS
Mid-Range Memory Chip-Select Register	A6H	MMCS
Memory Peripheral Chip-Select Register	A8H	MPCS

Upper Memory Chip-Select

The 80186 generates an upper memory chip-select output UCS(L) when accessing a memory block located at the top of memory. The upper limit of the memory block defined by this chip-select is always FFFFF. You can program the lower address limit of the block, permitting a block size from 1K bytes to 256K bytes. Table 2-5 shows the relationship between the starting address (lower address limit) selected and the size of the memory block. The upper memory block is defined in the UMCS Register. Figure 2-7 shows the bit format for the UMCS Register.

Table 2-5
UMCS Programming Values

Upper Memory Block Starting Address (Lower Address)	Upper Memory Block Size	UMCS Value (Assuming R0=R1=R2=0)	UMCS Bits 13-6
FFC00	1K	FFF8	11111111B
FF800	2K	FFB8	11111110B
FF000	4K	FF38	11111100B
FE000	8K	FE38	11111000B
FC000	16K	FC38	11110000B
F8000	32K	F838	11100000B
F0000	64K	F038	11000000B
E0000	128K	E038	10000000B
C0000	256K	C038	00000000B

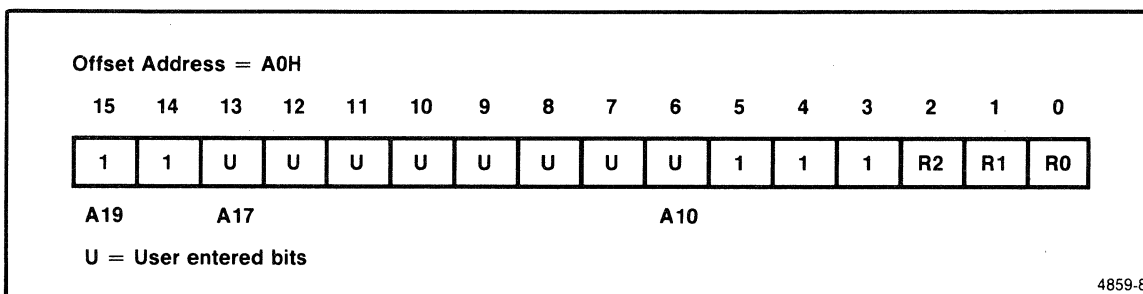


Figure 2-7. UMCS Register format.

The UMCS Register is at an offset address of A0H in the PCB, as shown in Figure 2-3. Table 2-5 also shows the legal programming values for the UMCS Register that define the lower address limit for the upper memory block. Any combination of bits 13 through 6 not shown in Table 2-5 is not allowed, and results in an undefined operation.

After a reset, the UMCS Register is programmed for a 1K-byte area. The register must be reprogrammed if you want a larger upper memory space. UMCS bits R2-R0 are used to specify a ready mode for the upper memory block. The setting of these bits is discussed later in this section.

Lower Memory Chip-Select

The 80186 generates a lower memory chip-select output LCS(L) when accessing a memory block located at the bottom of memory. The lower limit of the memory block defined by this chip-select is always 00000. You can program the upper address limit of the block, permitting a block size from 1K bytes to 256K bytes. Table 2-6 shows the relationship between the starting address (upper address limit) selected and the size of the memory block. The lower memory block is defined in the LMCS Register. Figure 2-8 shows the bit format for the LMCS Register.

Table 2-6
LMCS Programming Values

Lower Memory Block Starting Address (Upper Address)	Lower Memory Block Size	LMCS Value (Assuming R0=R1=R2=0)	LMCS Bits 13-6
003FF	1K	0038	00000000B
007FF	2K	0078	00000001B
00FFF	4K	00F8	00000011B
01FFF	8K	01F8	00000111B
03FFF	16K	03F8	00001111B
07FFF	32K	07F8	00011111B
0FFFF	64K	0FF8	00111111B
1FFFF	128K	1FF8	01111111B
3FFFF	256K	3FF8	11111111B

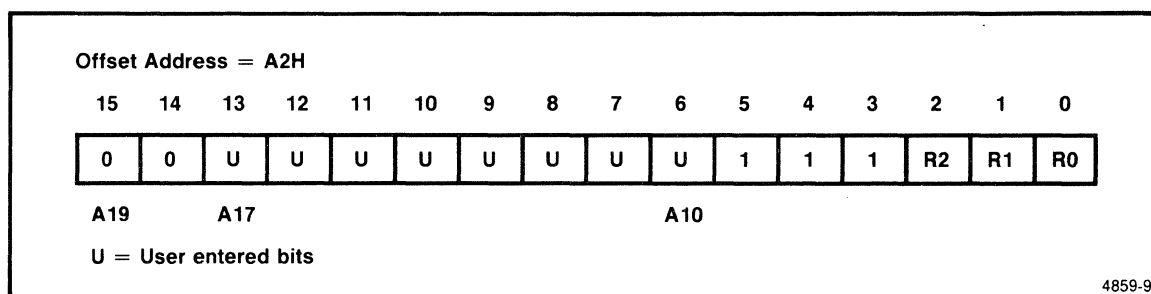


Figure 2-8. LMCS Register format.

The LMCS Register is at an offset address of A2H in the PCB, as shown in Figure 2-3. Table 2-6 also shows the legal programming values for the LMCS Register that define the upper address limit for the lower memory block. Any combination of bits 13 through 6 not shown in Table 2-6 is not allowed and results in an undefined operation. After a reset, the LMCS Register is undefined. The LCS(L) chip-select line is not active until the LMCS register is programmed. LMCS bits R2-R0 are used to specify a ready mode for the lower memory block. The setting of these bits is discussed later in this section.

Mid-Range Memory Chip-Selects

The 80186 generates four mid-range memory chip-select outputs MCS0(L)–MCS3(L) when accessing a user-relocatable memory block. This mid-range memory block is defined by two registers.

- The Memory/Peripheral Chip-Select Register (MPCS) determines the mid-range memory block size. You can program this memory block from 8K bytes to 512K bytes.
- The Mid-Range Memory Chip-Select Register (MMCS) determines the mid-range memory block starting address (lower address limit). The mid-range memory block can be located anywhere within the 1M-byte memory address space except for the memory blocks defined by the UMCS and LMCS registers.

The size of the mid-range memory block is determined by bits 14 through 8 in the MPCS Register. Table 2-7 shows the relationship between the total block size and bits 14 through 8 in the MPCS Register. Figure 2-9 shows the bit format for the MPCS Register.

Table 2-7
MPCS Programming Values

Total Mid-Range Memory Block Size	Individual Chip-Select Size	MPCS Bits 14–8 (M6–M0)
8K	2K	0000001B
16K	4K	0000010B
32K	8K	0000100B
64K	16K	0001000B
128K	32K	0010000B
256K	64K	0100000B
512K	128K	1000000B

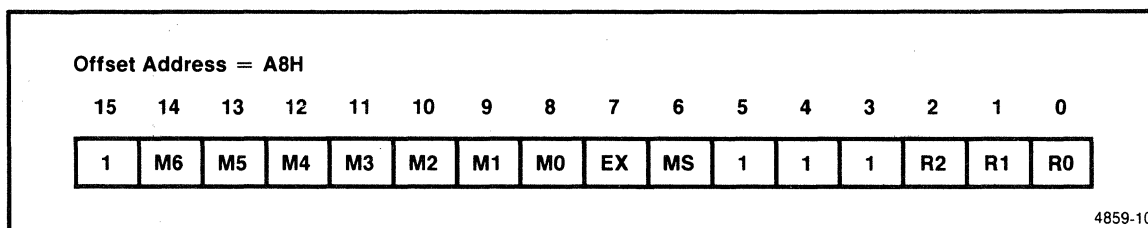


Figure 2-9. MPCS Register format.

The MPCS Register is at an offset address of A8H in the PCB as shown in Figure 2-3. Only one of the M6–M0 bits (bits 14 through 8) can be set at a time. If more than one bit is set, operation of the MCS0(L)–MCS3(L) output lines are unpredictable.

Each of the four chip-select lines is active for one fourth of the total mid-range memory block size. If the total block size is 64K, each chip-select is active for 16K of memory. MCS0(L) is active for the lower 16K portion of the block and MCS3(L) is active for the upper 16K portion of the block.

The mode of operation of the peripheral chip-selects is defined by bit 6 (MS bit) and bit 7 (EX bit) of the MPCS register. Refer to Figure 2-9. This register also sets the size of the mid-range memory block. Bit 6 is used to select whether the peripheral chip-selects are mapped into memory or I/O space. Bit 7 is used to select the function of chip-select lines PCS5(L) and PCS6(L). Table 2-9 shows the programming of these bits.

Table 2-9
Bits MS and EX Programming Values

Bit	Description
MS	1 = Peripherals mapped into memory space. 0 = Peripherals mapped into I/O space.
EX	0 = Five PCS lines, PCS0(L)-PCS4(L). A1 and A2 are both provided. 1 = Seven PCS lines, PCS0(L)-PCS6(L). A1 and A2 are not provided.

After a reset, the contents of both MPCS and PACS registers are undefined. None of the peripheral chip-select lines is active until both MPCS and PACS are programmed.

Ready Mode Generation

The 80186 generates an internal ready signal for each memory or peripheral chip-select line: UCS(L), LCS(L), MCS0(L)-MCS3(L), and PCS0(L)-PCS6(L). The number of wait states that are inserted for each memory or peripheral access can be programmed. Up to three wait states (0-3) can be programmed for all accesses to the area for which the chip-select line is active. You can also program the 80186 to ignore the external ready line for each chip-select range or to include the external ready line with the integrated ready generator.

The ready mode control consists of the three lower bits (R2-R0) of each chip-select register. The effect of the ready bits is shown in Table 2-10.

Table 2-10
Ready Mode Programming Bits

Bits			Number of Wait States Generated
R2	R1	R0	
0	0	0	0 wait states, external ready also used.
0	0	1	1 wait state inserted, external ready also used.
0	1	0	2 wait states inserted, external ready also used.
0	1	1	3 wait states inserted, external ready also used.
1	0	0	0 wait states, external ready ignored.
1	0	1	1 wait state inserted, external ready ignored.
1	1	0	2 wait states inserted, external ready ignored.
1	1	1	3 wait states inserted, external ready ignored.

The internal ready generator operates in parallel with the external ready signal, when the external ready is used ($R2 = 0$). For example, if the internal generator is set to insert two wait states but activity on the external ready signal inserts four wait states, the microprocessor inserts only four wait states, not six. The two wait states generated by the internal generator overlap the first two wait states generated by the external ready signal.

Bits R2-R0 of each chip-select control word specify the ready mode for the corresponding memory block, with the exception of the peripheral chip-selects. R2-R0 of the PACS Register set the ready mode for peripheral chip-selects PCS0(L)-PCS3(L). R2-R0 of the MPCS Register set the ready mode for peripheral chip-selects PCS4(L)-PCS6(L).

Timers

The 80186 has three internal 16-bit programmable timers. Timer 0 and Timer 1 are highly flexible and are connected to four external lines (two lines for each timer). Timer 0 and Timer 1 can be used to count external events, time external events, generate non-repetitive waveforms, etc. Timer 2 is an internal timer that is used for real-time coding and time delay applications. Timer 2 can also be used as a prescaler to the other two timers or as a DMA request source.

Timer Operation

The three internal timers are controlled by eleven 16-bit registers in the internal PCB. The timer registers' offset addresses and symbol names are shown in Table 2-11. The offset addresses are in relationship to the PCB's base address as shown in Figure 2-3.

The Count Register contains the current value of its timer. You can read or write to the register at any time regardless of whether the timer is running or not. The value of the Count Register is incremented for each timer event occurrence.

Each timer has a Maximum Count Register, which specifies the maximum count the timer reaches. After reaching the Maximum Count value, the Count Register resets to 0 during the same clock. The maximum count value is never stored in the Count Register.

Timer 0 and Timer 1 have a second Maximum Count Register, which permits the timers to alternately count between two different user-programmable Maximum Count values. If a single Maximum Count register is used, the timer output pin goes low for a single clock, two clock cycles after the maximum count value is reached. When using the dual Maximum Count register mode, the output line (TMR OUT 0 or TMR OUT 1) indicates which Maximum Count register is currently in use.

For Timer 0 or Timer 1, the RIU bit (bit 12) in the Mode/Control Word Register determines which Maximum Count register (A or B) is used for the comparison.

Each timer is serviced every fourth CPU-clock cycle and operates at speeds up to one-fourth the internal clock frequency. External clocking of the timers is also available at speeds up to one-fourth the internal clock (2 MHz for an 8 MHz CPU clock).

The timers have several programmable options that can be selected by setting the Mode/Control Word Register of each timer:

- All timers can be set to halt or continue on a terminal count.
- Timer 0 and Timer 1 can select between internal and external clocks, alternate between Maximum Count registers, and be retrigger on external events.
- The timers may be programmed to generate an interrupt on a terminal count.

Table 2-11
Timer Register Offset Addresses and Symbol Names

Register Name	Offset Address	Symbol Name
Timer 0 Registers:		
Count Register	50H	T0CNR
Maximum Count A Register	52H	T0MXA
Maximum Count B Register	54H	T0MXB
Mode/Control Word Register	56H	T0MCW
Timer 1 Registers:		
Count Register	58H	T1CNR
Maximum Count A Register	5AH	T1MXA
Maximum Count B Register	5CH	T1MXB
Mode/Control Word Register	5EH	T1MCW
Timer 2 Registers:		
Count Register	60H	T2CNR
Maximum Count A Register	62H	T2MXA
Maximum Count B Register	Not Present	
Mode/Control Word Register	66H	T2MCW

Timer Mode/Control Word Register

The Mode/Control Word Register for each timer permits you to program the specific operating mode or check the current programmed status of the three integrated timers. Figure 2-12 shows the function of each bit in the Mode/Control Word Register.

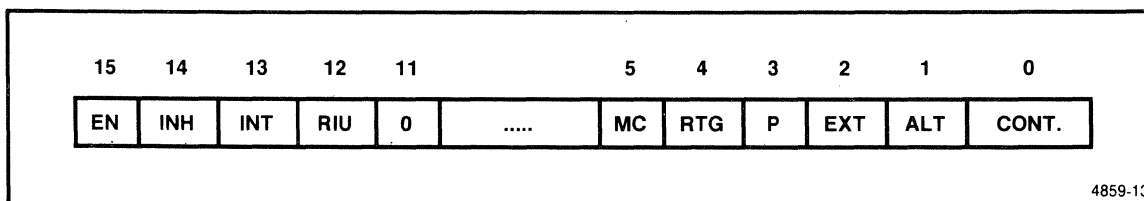


Figure 2-12. Timer Mode/Control Word Register format.

The following paragraphs define the bits in the Timer Mode/Control Word Register shown in Figure 2-12. These bits, are defined from an operational viewpoint because of their interactions.

- Bit 1 **ALT.** The ALT bit determines which of the two Maximum Count registers is used for a count comparison with the Count Register. If this bit is low (ALT = 0), Register A for that timer is always used. If this bit is high (ALT = 1), the dual Maximum Count mode is established.

When in the dual Maximum Count mode, the count comparison alternates between Maximum Count A Register and Maximum Count B Register. When the maximum count on one register is reached, the comparison switches to the other register. The alternating comparison permits you to change one Maximum Count register while the other is being used. This provides a method of generating non-repetitive waveforms.

The ALT bit also determines the function of the timer output signal (TMR OUT 0 or TMR OUT 1). If ALT is 0, the output signal goes low for one clock cycle after the maximum count is reached. If ALT is 1, the output signal reflects the current Maximum Count register being used (0 = Register B or 1 = Register A).

- Bit 0 **CONT.** Setting the continuous bit high causes the associated timer to run continuously. If this bit is cleared (CONT = 0), the timer halts when a maximum count is reached. If the CONT bit = 0 and the ALT bit = 1, the timer counts to the Maximum Count A Register value and then resets, counts to the Maximum Count B Register value and resets, and then halts.

- Bit 2 **EXT.** The external bit selects between internal and external clocking for the timer. The external signal may be asynchronous with respect to the 80186 internal clock. If this bit is set (EXT = 1), the timer counts low-to-high transitions on the input signal (TMR IN 0 or TMR IN 1). If this bit is cleared (EXT = 0), it counts an internal clock while using the input signal for control. In this mode, the function of the input signal is defined by the RTG bit.

- Bit 3 **P.** The prescaler bit is ignored unless internal clocking is selected (EXT = 0). If the P bit is 0, the timer will count at one-fourth the internal CPU clock rate. If the P bit is 1, the output of Timer 2 is used as a clock for the timer. You must initialize and start Timer 2 to provide the prescaled clock.

- Bit 4 **RTG.** The retrigger bit is active only for internal clocking (EXT = 0). When internal clocking is selected, the RTG bit determines the control function provided by the input signal (TMR IN 0 or TMR IN 1).

If RTG = 0, the input signal level gates the internal clock on and off. If the input signal is high, the timer counts. If the input signal is low, the timer holds its value and stops counting.

When RTG = 1, low-to-high transitions in the input signal are detected. The first input signal's low-to-high transition starts the timer running. The timer value is cleared on the first internal clock cycle and incremented for each succeeding internal clock cycle. Additional low-to-high transitions on the input signal reset the timer to 0 and the timer starts counting its internal clock again. If CONT = 0, when the timer reaches its maximum count the EN bit (bit 15) is cleared, inhibiting further timer activity.

- Bit 15 **EN.** The enable bit provides programmer control over the timer's RUN/HALT status. When set ($EN = 1$), the timer is enabled to count subject to the input signal constraints in the internal clock mode (refer to bit 2). When this bit is cleared ($EN = 0$), the timer is inhibited from counting. The state of the input signal is ignored while EN is 0. If $CONT$ is 0, the EN bit is automatically cleared when the timer reaches its maximum count.
- Bit 14 **INH.** The inhibit bit permits selective updating of the enable (EN) bit. If INH is high during a write to the Mode/Control Word Register, the state of the EN bit is modified by the write operation. If INH is low during a write to the Mode/Control Word Register, the state of the EN bit is not affected by the write operation. The INH bit is not stored; it is always 0 on a read to the Mode/Control Word Register.
- Bit 13 **INT.** If this bit is set ($INT = 1$), an interrupt request is generated each time the Maximum Count Register reaches its maximum count. If the timer is configured for dual Maximum Count mode, an interrupt is generated each time Register A or Register B reaches its maximum count. If this bit is cleared ($INT = 0$) after an interrupt request is generated, but before the pending interrupt is serviced, the interrupt request is still active. (The interrupt request is latched in the Interrupt Controller.)
- Bit 5 **MC.** The maximum count bit is set whenever the timer reaches its maximum count value. If the timer is configured for dual Maximum Count mode, this bit is set each time Register A or Register B reaches its maximum count. This bit is set regardless of the timer's INT bit (interrupt enable bit). The MC bit lets you monitor the timer's status through software instead of through interrupts.
- Bit 12 **RIU.** The register-in-use bit indicates which Maximum Count register is currently being used for comparison with the Count Register value. If $RIU = 0$, Register A is in use. If $RIU = 1$, Register B is in use. You cannot write to the RIU bit. Its value is not affected when you write to the Mode/Control Word Register. This bit is always cleared when the ALT bit is 0.

Not all Mode/Control Word bits are programmable for Timer 2. The following bits are hardwired for Timer 2:

- $ALT = 0$
- $EXT = 0$
- $P = 0$
- $RTG = 0$
- $RIU = 0$

Count Registers

The Count Register for each timer is a 16-bit register. The 80186 can read from or write to the current contents of this register at any time. If the 80186 writes to the register while the timer is counting, the new value takes effect in the current count cycle.

Maximum Count Registers

Timer 0 and Timer 1 have two Maximum Count registers, Maximum Count A and Maximum Count B. Timer 2 has a single Maximum Count register, Maximum Count A. The value in the Maximum Count registers contain the number of events the timer counts. When Timer 0 and Timer 1 are configured for dual Maximum Count mode, the counting alternates between the Maximum Count A Register and Maximum Count B Register values. A timer resets during a comparison between the current timer count value in the Count Register and the maximum value in the Maximum Count register. When these two values are equal, the timer resets to 0 (Count Register is reset to 0).

If the current value in the Count Register is changed to be above the maximum count value, or if the maximum count value is changed to be below the current count value, the timer does not reset to 0. The timer instead counts to the Count Register's maximum value (16 bits), "wraps around" to 0, then counts until the maximum count value is reached.

Timers and Reset

When a reset occurs, the Timers perform the follow:

- All EN (Enable) bits (bit 15 in the Timer Mode/Control Word registers) are reset, preventing timer counting.
- All ALT (select) bits (bit 1 in the Timer Mode/Control Word registers) are reset to 0. This selects Maximum Count A Register, which results in the output signals (TMR OUT 0 and TMR OUT 1) going high on the reset.

Interrupt Controller

The 80186 receives interrupts from a number of sources, both internal and external. The internal Interrupt Controller merges these requests on a priority basis for individual service by the CPU.

Internal interrupt sources (Timers and DMA channels) are disabled by their own control registers or by mask bits within the Interrupt Controller. The Interrupt Controller has its own control registers that set the operation mode for the controller.

The Interrupt Controller resolves priority among requests that are pending simultaneously. Nesting is provided so that interrupt service routines for lower priority interrupts may be interrupted by higher priority interrupts.

The 80186 Interrupt Controller has two basic operation modes: NON-RMX mode and RMX mode. The RMX mode is a special 8086 compatibility mode that allows the use of the 80186 within the 8086 operating system interrupt structure. The internal 80186 Interrupt Controller is set in this mode by setting bit 14 high in the PCB's Relocation Register as shown in Figures 2-3 and 2-4. In the RMX mode, the internal 80186 Interrupt Controller functions as a slave controller to the external master controller. Special initialization software must be included to properly set up the interrupt source priority level for the 80186 Interrupt Controller in RMX mode.

NON-RMX Mode Operation

The Interrupt Controller services five dedicated external interrupt lines. One interrupt line is dedicated to NMI (non-maskable interrupt). The function of the other four external interrupt lines, INTO(H) through INT3(H), determines the Interrupt Controller's operating mode:

- Fully Nested Mode** All four external interrupt lines, are used as direct interrupt requests.
- Cascade Mode** The four external interrupt lines are configured into interrupt requests and interrupt acknowledge signal pairs. INTO(H) and INT1(H) serve as direct interrupt requests. INT2(H)/INTA0(L) and INT3(H)/INTA1(L) serve as dedicated interrupt acknowledge signals.

NON-RMX Mode Interrupt Controller Registers. The Interrupt Controller registers' offset addresses and symbol names are shown in Table 2-12. The offset addresses are in relationship to the PCB's base address as shown in Figure 2-3. Fifteen Interrupt Controller registers are used in the NON-RMX mode. You can read or write to all registers unless otherwise specified.

Table 2-12
Interrupt Controller Registers (NON-RMX Mode)

Register Name	Offset Address	Symbol Name
EOI Register	22H	ICRER ^a
Poll Register	24H	ICRPR ^b
Poll Status Register	26H	ICRPS ^b
Mask Register	28H	ICRMR
Priority Mask Register	2AH	ICRPM
In-Service Register	2CH	ICRIN
Interrupt Request Register	2EH	ICRIR
Interrupt Status Register	30H	ICRIS
Timer Control Register	32H	ICRTC
DMA 0 Control Register	34H	ICRD0
DMA 1 Control Register	36H	ICRD1
INT0 Control Register	38H	ICRI0
INT1 Control Register	3AH	ICRI1
INT2 Control Register	3CH	ICRI2
INT3 Control Register	3EH	ICRI3

^a Write only register.

^b Read only register.

In-Service Register. The In-Service Register bit format is shown in Figure 2-13. You can read or write to this register. The In-Service Register contains the In-Service (IS) bits for each of the interrupt sources: one Timer source, two DMA sources, and four external interrupt sources. A particular IS bit is set to indicate that an interrupt source service routine is in progress. When an IS bit is set, the Interrupt Controller does not generate interrupts to the CPU when the Interrupt Controller receives interrupt requests from devices with a lower programmed priority level.

The bits in the In-Service Register shown in Figure 2-13 have the following functions:

- Bit 0 **TMR.** The timer bit is the IS bit for all three internal timers.
- Bits 2 and 3 **D0 and D1.** These two bits are the IS bits for the two DMA channels.
- Bits 4–7 **I0–I3.** These four bits are the IS bits for the four external interrupt lines, INT0(H)–INT3(H).

The IS bit is set when the processor acknowledges an interrupt request either by a interrupt acknowledge, or by reading the Poll Register. The IS bit is reset at the end of the interrupt service routine by an end-of-interrupt (EOI) command issued by the CPU.

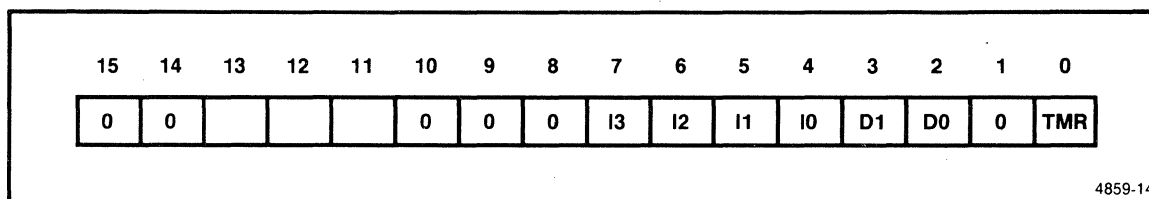


Figure 2-13. In-Service, Interrupt Request, and Mask register formats.

NOTE

In this section the bit format figures for all Interrupt Controller registers show some bits with no values called out (blank boxes). These bits are "don't cares."

Interrupt Request Register. The Interrupt Request Register contains the status of interrupt request bits from internal interrupt sources: one Timer source and two DMA channel sources. The Interrupt Request Register bit format is also shown in Figure 2-13. You can read this register to determine the status of the interrupt request bits. The TMR bit (bit 0) is the logical OR of all three timer interrupt requests. Bits D0 and D1 (bits 2 and 3) are the interrupt request bits for the two DMA channels.

The interrupt request bits from internal interrupt sources (Timers or DMA channels) are set when the internal interrupt requests arrive at the Interrupt Controller and are reset when the microprocessor acknowledges the interrupt request.

The states of the external interrupt lines are also shown by the interrupt request bits I0–I3 (bits 4–7) in the Interrupt Request Register. The states of the external interrupt lines are not a stored condition in the Interrupt Controller. Because the line states are not stored, you cannot write to the external interrupt request bits (bits 4–7) in the Interrupt Request Register.

The external interrupt request bits (I0–I3) show exactly when an interrupt request is given to the Interrupt Controller. If the edge-triggered mode is selected, an interrupt request bit in the Interrupt Request Register is high only after an inactive-to-active (low-to-high) transition on an external interrupt line.

Mask Register. The Mask Register is a 16-bit register and contains a mask bit for each internal and external interrupt source. The Mask Register bit format is also shown in Figure 2-13. Setting a bit in the Mask Register that corresponds to a particular interrupt source masks the interrupt source from generating interrupts to the Interrupt Controller. The mask bits in the Mask Register are the same bits (MSK bits) that are in the individual control registers (discussed later in this section). Programming the mask bits in the Mask Register also changes the MSK bits in the individual control registers, and vice versa.

Priority Mask Register. The Priority Mask Register is used to mask all interrupts below a specified interrupt priority level. The Priority Mask Register bit format is shown in Figure 2-14. The priority level specified by the lower three bits (PRM2–PRM0) in this register inhibits interrupts with priorities lower (a higher priority number) than that specified. For example, if priority level 4 is set into the lower three bits (100), the register masks all interrupts of priority levels 5 (101), 6 (110), and 7 (111). The lower three bits are set to priority level 7 (111) upon a reset, which unmask all interrupts.

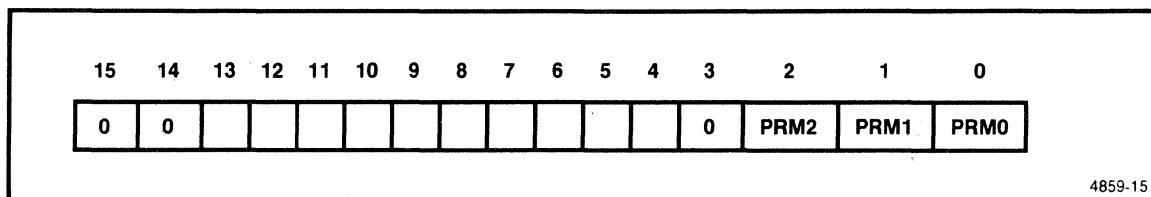


Figure 2-14. Priority Mask Register format.

Interrupt Status Register. The Interrupt Status Register provides general status information on the Interrupt Controller. The Interrupt Status Register bit format is shown in Figure 2-15. The bits in the Interrupt Status Register have the following functions:

Bit 15 **DHLT.** Setting the DMA Halt Transfer bit halts all DMA transfers. This bit is automatically set whenever a non-maskable interrupt occurs, and is reset when an IRET instruction is executed. This bit's main purpose is to permit prompt service of all non-maskable interrupts. This bit can also be set by the CPU.

Bits 2, 1, and 0 **IRT2–IRT0.** The lower three bits in the Interrupt Status Register show the status of the individual timer interrupt request bits. Bits IRT0 through IRT2 correspond to Timer 0 through Timer 2. The three bits distinguish between the timer interrupts.

Recall that the TMR bit (bit 0) in the Interrupt Request Register is the logical OR function of all timer interrupt requests. Setting any one of the three lower bits in the Interrupt Status Register initiates a timer interrupt request to the Interrupt Controller and sets the TMR bit in the Interrupt Request Register.

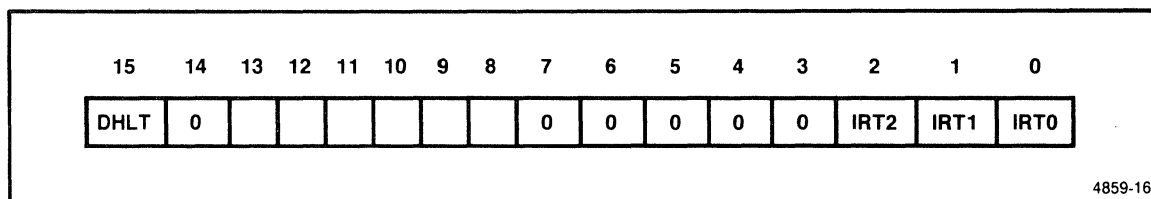


Figure 2-15. Interrupt Status Register format.

Timer and DMA Control Registers. Three 16-bit registers control all the internal interrupt sources:

- Timer Control Register
- DMA 0 Control Register
- DMA 1 Control Register

The bit format for these three registers is shown in Figure 2-16. The three lower bits PR2-PR0 represent the programmable priority level of the associated interrupt source. The programmable interrupt levels are 0-7. The MSK bit (bit 3) inhibits an interrupt request from the associated internal interrupt source. The MSK bits in the individual control registers are the same bits that appear in the Mask Register (refer to Figure 2-13).

- Timer Control Register MSK bit is the same as the Mask Register TMR bit (bit 0).
- DMA 0 Control Register MSK bit is the same as the Mask Register D0 bit (bit 2).
- DMA 1 Control Register MSK bit is the same as the Mask Register D1 bit.

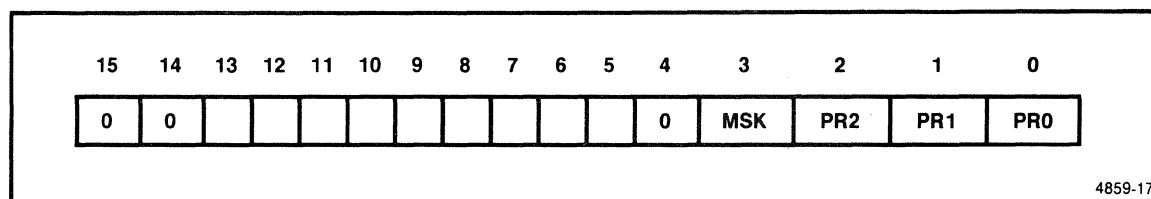


Figure 2-16. Timer/DMA Control register formats.

External Interrupt Control Registers. Four 16-bit registers control all the external interrupt sources:

- INT0 Control Register
- INT1 Control Register
- INT2 Control Register
- INT3 Control Register

The bit format for the INT0 and INT1 Control registers is shown in Figure 2-17. The bit format for the INT2 and INT3 Control registers is shown in Figure 2-18. In cascade mode or special fully nested mode, the control registers INT2 and INT3 are not used. The bits in the various external interrupt (INT) control registers have the following functions:

- Bits 2-0** **PR2-PR0.** The lower three bits in all four registers specify the priority of the external source interrupt. The highest priority is 0 (000) and the lowest priority is 7 (111).
- Bit 4** **LTM.** The level-trigger mode bit in each register specifies the input triggering level of the external interrupt source: 1 = level-triggered input; 0 = edge-triggered input. The external interrupt source signals are active high. For level-triggered mode, an interrupt is generated when the external interrupt line is high. For edge-triggered mode, an interrupt is generated only when the active high level is preceded by a low-to-high transition on the external interrupt input line. For both modes, the active high must remain high until the interrupt is acknowledged.
- Bit 3** **MSK.** The mask bit in each register inhibits an interrupt request from the associated external interrupt source. The MSK bits in the individual control registers are the same bits that appear in the Mask Register (refer to Figure 2-13).
- INT0 Control Register MSK bit is the same as the Mask Register I0 bit (bit 4).
 - INT1 Control Register MSK bit is the same as the Mask Register I1 bit (bit 5).
 - INT2 Control Register MSK bit is the same as the Mask Register I2 bit (bit 6).
 - INT3 Control Register MSK bit is the same as the Mask Register I3 bit (bit 7).
- Bit 5** **C.** (For Figure 2-17 only.) The cascade mode bit in the interrupt control registers INT0 and INT1 specifies cascade mode or fully nested mode: 1 = cascade mode; 0 = fully nested mode (direct mode).
- Bit 6** **SFNM.** (For Figure 2-17 only.) The special fully nested mode bit in the interrupt control registers INT0 and INT1 specifies the mode of operation: 1 = SFNM (special fully nested mode).

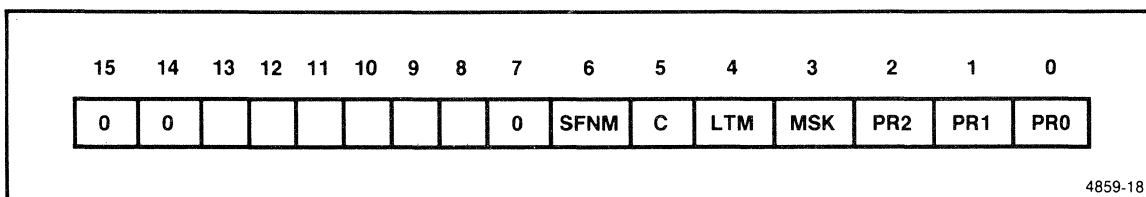


Figure 2-17. INT0 and INT1 Control register formats.

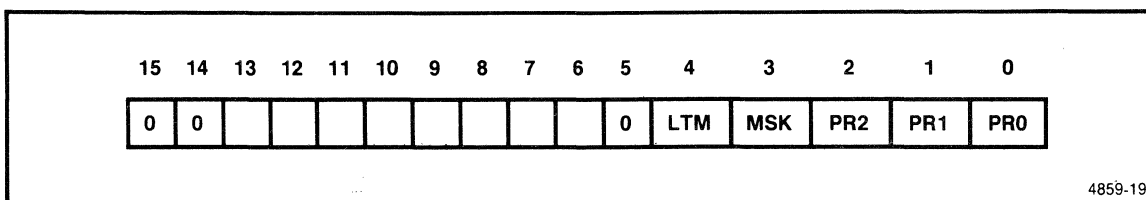


Figure 2-18. INT2 and INT3 Control register formats.

EOI Register. The EOI (end-of-interrupt) Register is a command register that can only be written to by the 80186. The EOI Register bit format is shown in Figure 2-19. When the 80186 writes to this register, an EIO command is initiated which resets the IS bit for a specific interrupt. The bits in the EOI Register have the following functions:

Bits 4-0 **S4-S0.** The lower five bits of the EOI Register contain encoded information that specifies the interrupt source vector type for a specific interrupt. The interrupt source vector types are shown in Table 2-13. The vector type set into these bits resets the associated IS bit in the In-Service Register (refer to Figure 2-13). For example, to reset the IS bit for DMA Channel 0, these bits should be set to 01010. The vector type for DMA Channel 0 is 10 (decimal notation).

NOTE

Vector type 8 (01000) for Timer 0 should be written to the EOI Register to reset the single IS bit for any of the three timers. The TMR bit (bit 0 in the In-Service Register) is the IS bit for all three timers.

Bit 15 **SPEC/NSPEC.** The specific/nonspecific bit determines the EOI command type: 1 = nonspecific and 0 = specific.

Table 2-13
80186 Interrupt Vectors

Interrupt Name	Vector Type ^a	Default Priority ^b
Timer 0 Interrupt	8	0A ^c
Timer 1 Interrupt	18	0B ^c
Timer 2 Interrupt	19	0C ^c
Reserved	9	1
DMA 0 Interrupt	10	2
DMA 1 Interrupt	11	3
INT0 Interrupt	12	4
INT1 Interrupt	13	5
INT2 Interrupt	14	6
INT3 Interrupt	15	7

^a The vector type is a decimal notation.

^b The default priorities for the interrupt sources are used only if the user does not program each source into a unique priority level.

^c All three timers constitute one source request to the Interrupt Controller. The timer interrupts have priorities assigned, but have the same default priority level with respect to all other interrupt sources. For example, timer Priority 0A is higher priority than 0B. Each Timer interrupt has a separate vector type number.

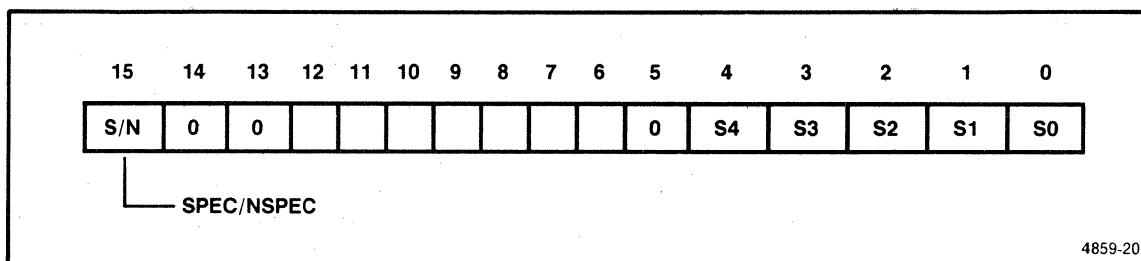


Figure 2-19. EOI Register format.

Poll and Poll Status Registers. The Poll Register and Poll Status Register contain polling information. Both registers contain the same information, but have slightly different functions. The Poll and Poll Status register bit formats are shown in Figure 2-20. Both registers can only be read.

Reading the Poll Register initiates a software poll, which sets the IS bit of the highest pending priority interrupt. Reading the Poll Status Register provides the status of the pending interrupts and does not set the IS bit of the highest pending priority interrupt. The bits in the Poll Register and Poll Status Register have the following functions:

- Bits 4-0** **S4-S0.** The lower five bits of the Poll and Poll Status registers contain encoded information that specifies the interrupt source vector type for a specific interrupt. The interrupt source vector types are shown in Table 2-13. The vector type indicated by these bits is only valid when the INTREQ bit (bit 15) is 1.
- Bit 15** **INTREQ.** The interrupt request bit indicates if an interrupt request is pending: 1 = interrupt request is pending and 0 = no interrupt request is pending.

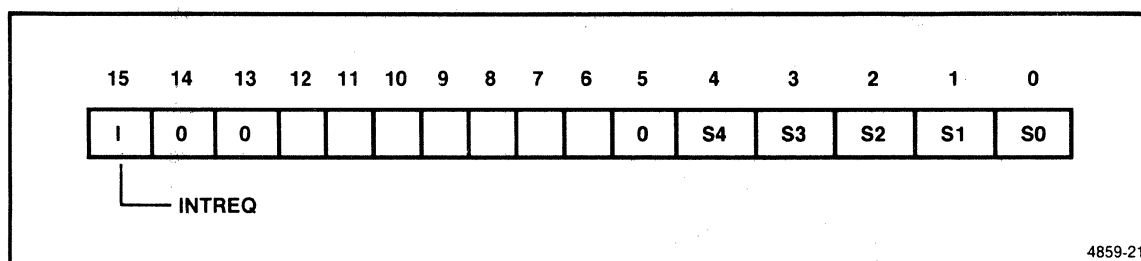


Figure 2-20. Poll and Poll Status register formats.

RMX 8086 Compatibility Mode

The RMX mode permits 8086/80186 compatibility. The interrupt circuitry of an 8086 consists of one master and multiple slave 8529As configured in cascade operating mode. When RMX mode is used, the internal 80186 Interrupt Controller is used as a slave controller to an external Master Interrupt Controller. The internal 80186 resources are monitored through the internal Interrupt Controller. The external controller functions as the system Master Interrupt Controller.

Upon reset, the 80186 Interrupt Controller is in the NON-RMX operating mode. To set the controller in the RMX mode, set bit 14 of the Relocation Register to 1 (refer to Figure 2-4).

In the RMX operating mode, the 80186 Interrupt Controller no longer accepts external interrupt sources. This is due to pin limitations in the 80186 microprocessor caused by the need to interface to an external Master Interrupt Controller. However, sufficient internal 80186 Interrupt Controller inputs exist for each timer and DMA channel. In the RMX mode, each internal interrupt source (three timers and two DMA channels) has its own mask bit, IS bit, and control word.

The RMX mode requires that the peripherals be assigned fixed priority levels. This requirement is incompatible with the normal operation of the 80186 Interrupt Controller. The software initialization must program the proper interrupt priority levels for each internal interrupt source. The required priority levels for the internal interrupt sources in RMX mode are shown in Table 2-14. The level assignments must remain fixed while in the RMX operating mode.

Table 2-14
RMX Mode Internal Interrupt Source Priority Level

Priority Level	Interrupt Source
0	Timer 0
1	(Reserved)
2	DMA 0
3	DMA 1
4	Timer 1
5	Timer 2

RMX Mode Interrupt Controller Registers. The Interrupt Controller registers' offset addresses and symbol names are shown in Table 2-15. The offset addresses are in relationship to the PCB's base address (refer to Figure 2-3). There are 12 Interrupt Controller registers used in the RMX mode. Some of these registers are also used in the NON-RMX mode, but the register bit functions and some register symbol names are different. You can read or write to all registers unless otherwise specified.

Table 2-15
Interrupt Controller Registers (RMX Mode)

Register Name	Offset Address	Symbol Name
Interrupt Vector Register	20H	ICRIV
Specific EOI Register	22H	ICRSE ^a
Mask Register	28H	ICRMR
Priority Level Mask Register	2AH	ICRPM
In-Service Register	2CH	ICRIN
Interrupt Request Register	2EH	ICRIR
Interrupt Status Register	30H	ICRIS
Level 0 Control Register (Timer 0)	32H	ICRL0
Level 2 Control Register (DMA 0)	34H	ICRL2
Level 3 Control Register (DMA 1)	36H	ICRL3
Level 4 Control Register (Timer 1)	38H	ICRL4
Level 5 Control Register (Timer 2)	3AH	ICRL5

^a Write only register.

EOI Register. The EOI (end-of-interrupt) Register is a command register that can only be written to by the microprocessor. The EOI Register bit format is shown in Figure 2-21. When the 80186 writes to this register, an EOI command is initiated that resets the IS bit for a specific interrupt. The bits in the EOI Register have the following function:

Bits 2-0 **L2-L0.** The lower three bits contain the encoded value that specifies the priority level of the IS bit being reset.

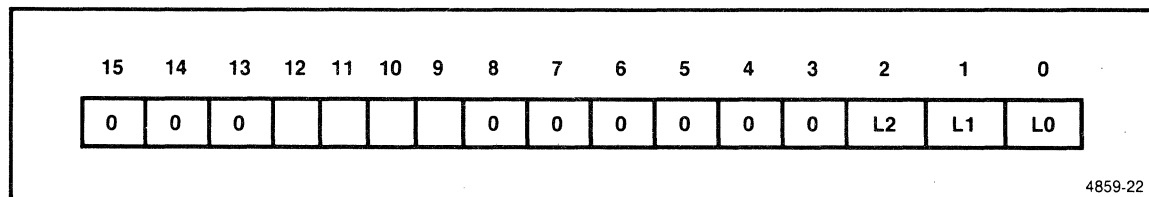


Figure 2-21. EOI Register format.

In-Service Register. The In-Service Register contains the IS (in-service) bit for each of the internal interrupt sources: three timer sources and two DMA sources. You can read or write to this register. The internal source IS bit is set when the 80186 acknowledges the internal interrupt's request. The In-Service Register bit format is shown in Figure 2-22. The bits in the In-Service Register have the following functions:

Bits 5, 4, and 0 **TMR2–TMR0.** These timer bits correspond to the three internal timers.

Bits 3 and 2 **D1 and D0.** These DMA bits correspond to the two DMA channels.

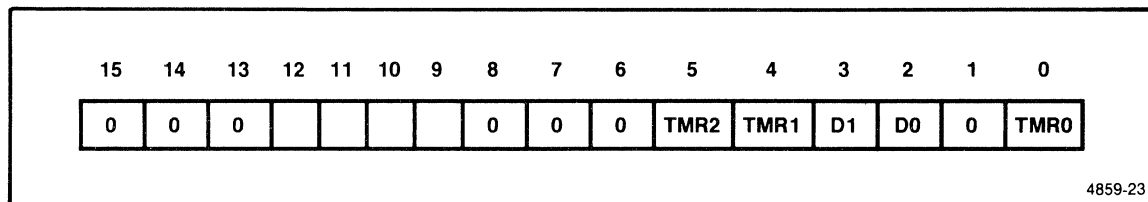


Figure 2-22. In-Service, Interrupt Request, and Mask register formats.

Interrupt Request Register. The Interrupt Request Register contains the status of internal peripherals that have interrupt requests pending. The Interrupt Request Register bit format is also shown in Figure 2-22. The interrupt request bits are set when an interrupt request arrives from an internal source, and are reset when the 80186 acknowledges the request. The bits in the Interrupt Request Register have the same functions as the bits in the In-Service Register.

Mask Register. The Mask Register contains a mask bit for each internal interrupt source. The Mask Register bit format is also shown in Figure 2-22. The bits in the Mask Register have the same functions as the bits in the In-Service Register. If you set a mask bit in the Mask Register that corresponds to a particular interrupt source, you mask the internal interrupt source from generating interrupts to the Interrupt Controller. The mask bits in the Mask Register are the same bits (MSK) that are in the individual control registers (discussed later in the section). Changing the state of a mask bit also changes the MSK bit in the individual control register. If you change the state of a MSK bit in the individual control register, you also change the state of the mask bit.

Timer/DMA Control Registers. Five 16-bit registers contain the control words for all the internal interrupt sources:

- Level 0 Control Register (Timer 0)
- Level 2 Control Register (DMA 0)
- Level 3 Control Register (DMA 1)
- Level 4 Control Register (Timer 1)
- Level 5 Control Register (Timer 2)

The bit format for these five registers is shown in Figure 2-23. The bits in the five Control Registers have the following functions:

- Bits 2-0** **PR2-PR0.** The three lower bits in each register specify the priority level for the associated internal interrupt source. Each source must be programmed for a specific level as shown in Table 2-14.
- Bit 3** **MSK.** The mask bit inhibits an interrupt request from the associated internal interrupt source. The MSK bit in the individual control registers is the same bit that appears in the Mask Register: (Refer to Figure 2-22.)
- Level 0 Control Register (Timer 0) MSK bit is the same as the Mask Register TMR0 bit (bit 0).
 - Level 2 Control Register (DMA 0) MSK bit is the same as the Mask Register D0 bit (bit 2).
 - Level 3 Control Register (DMA 1) MSK bit is the same as the Mask Register D1 bit (bit 3).
 - Level 4 Control Register (Timer 1) MSK bit is the same as the Mask Register TMR1 bit (bit 4).
 - Level 5 Control Register (Timer 2) MSK bit is the same as the Mask Register TMR2 bit (bit 5).

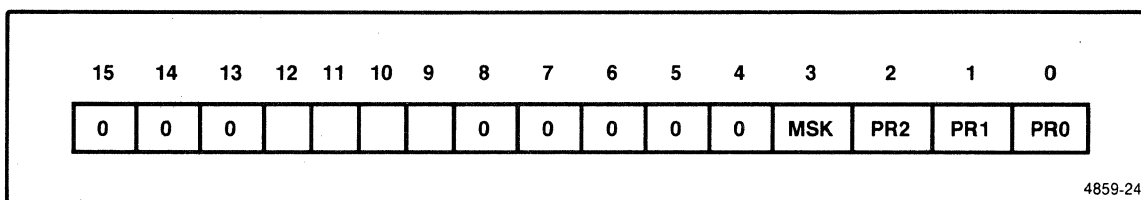


Figure 2-23. Timer/DMA Control register formats.

Interrupt Vector Register. The Interrupt Vector Register specifies the upper five bits of the Interrupt Vector address. The Interrupt Controller provides the lower three bits of the Interrupt Vector address. The lower three bits are determined by the priority level of the interrupt request. These bits are the same as the lower three bits in the associated Control Register. The Interrupt Vector Register bit format is shown in Figure 2-24. The bits in the Interrupt Vector Register have the following function:

- Bits 7-3** **T4-T0.** These bits are the upper five bits of the Interrupt Vector's address.

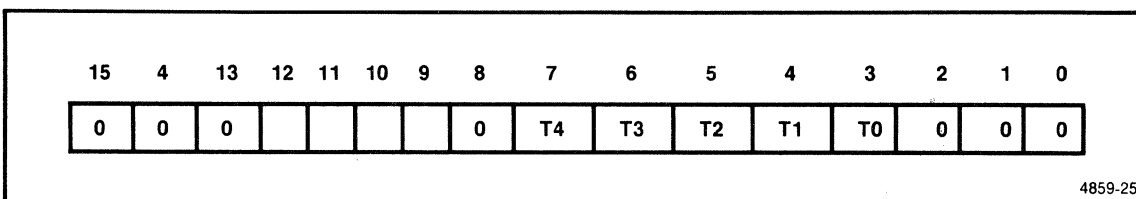


Figure 2-24. Interrupt Vector Register format.

Priority Level Mask Register. The lower three bits of the Priority Level Mask Register specifies the lowest priority level interrupt that is serviced. The Priority Level Mask Register bit format is shown in Figure 2-25. The bits in this register have the following function:

Bits 2-0 **M2-M0.** The lower three bits specify a priority level value. All interrupt requests with priority levels below this value are masked. These interrupt requests are inhibited.

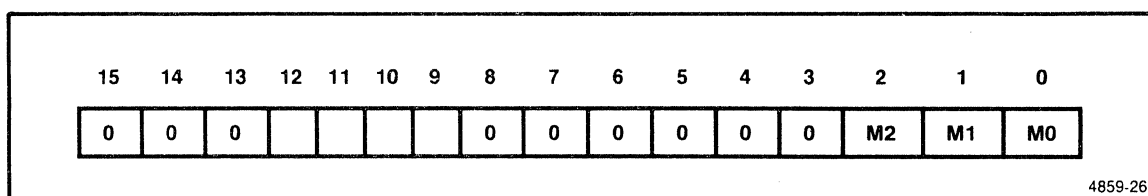


Figure 2-25. Priority Level Mask Register format.

Interrupt Controller and Reset

When a reset occurs, the Interrupt Controller performs the following actions:

1. All SFNM bits (bit 6 in the INT0 and INT1 Control registers) are reset to 0, specifying Fully Nested mode.
2. All PR bits (bits 2-0 in the individual control registers) are set to 1. This places all interrupt sources at the lowest priority level (111).
3. All LTM bits (bit 4 in the individual external control registers) are reset to 0. This results in the edge-triggered mode, in which interrupts are generated when a low-to-high transition is on the line.
4. All IS bits in the In-Service Register are reset to 0, indicating that no interrupt is being serviced.
5. All Interrupt Request bits in the Interrupt Request Register are reset to 0, indicating that no interrupt source is requesting an interrupt.
6. All MSK bits are set to 1 (mask), indicating that all interrupts are masked.
7. All C (Cascade) bits (bit 5 in the INT0 and INT1 Control Registers) are reset to 0 (non-cascade or fully nested mode).
8. All PRM (Priority Mask) bits (bits 2-0 in the Priority Mask Register) are set to 1, indicating no priority levels are masked.
9. Bit 14 in the Relocation Register is reset to 0, initializing the microprocessor to the NON-RMX mode.

Peripheral Control Block Register Summary

A summary of all PCB register symbol names used in the preceding PCB register descriptions are shown in Tables 2-16 and Tables 2-17. The DMA, Chip-Select, and Timer registers have the same symbol names for both RMX and NON-RMX modes. The register symbol names for these registers are shown in Table 2-16. Some of the Interrupt Controller register symbol names are not the same for RMX and NON-RMX modes. The register symbol names for the Interrupt Controller registers (both RMX and NON-RMX modes) are shown in Table 2-17.

Table 2-16
Symbol Names for DMA, Chip-Select, and Timer Registers

Symbol Name	Description
RELREG	Relocation Register
DMA0CW	DMA Registers: DMA Channel 0 Control Word Register
DMA0DP	DMA Channel 0 Destination Pointer Register
DMA0SP	DMA Channel 0 Source Pointer Register
DMA0TC	DMA Channel 0 Transfer Count Register
DMA1CW	DMA Channel 1 Control Word Register
DMA1DP	DMA Channel 1 Destination Pointer Register
DMA1SP	DMA Channel 1 Source Pointer Register
DMA1TC	DMA Channel 1 Transfer Count Register
LMCS	Chip-Select Registers: Lower Memory Chip Select Register
MMCS	Mid-Range Memory Chip Select Register
MPCS	Memory Programming Chip Select Register
PACS	Peripheral Address Chip Select Register
UMCS	Upper Memory Chip Select Register
T0CNR	Timer Registers: Timer 0 Count Register
T0MCW	Timer 0 Mode/Control Word Register
T0MXA	Timer 0 Maximum Count A Register
T0MXB	Timer 0 Maximum Count B Register
T1CNR	Timer 1 Count Register
T1MCW	Timer 1 Mode/Control Word Register
T1MXA	Timer 1 Maximum Count A Register
T1MXB	Timer 1 Maximum Count B Register
T2CNR	Timer 2 Count Register
T2MCW	Timer 2 Mode/Control Word Register
T2MXA	Timer 2 Maximum Count A Register

Table 2-17
Symbol Names for Interrupt Control Registers

Symbol Name	Description
Interrupt Control Registers: NON-RMX Mode	
ICRD0	ICR DMA Channel 0 Register
ICRD1	ICR DMA Channel 1 Register
ICRER ^b	ICR EOI Register
ICRIN ^a	ICR In-Service Register
ICRIR ^a	ICR Interrupt Request Register
ICRIS ^a	ICR Interrupt Status Register
ICRI0	ICR INT0 Control Register
ICRI1	ICR INT1 Control Register
ICRI2	ICR INT2 Control Register
ICRI3	ICR INT3 Control Register
ICRMR ^a	ICR Mask Register
ICRPM ^a	ICR Priority Mask Register
ICRPR ^c	ICR Poll Register
ICRPS ^c	ICR Poll Status Register
ICRTC	ICR Timer Control Register
Interrupt Control Registers: RMX Mode	
ICRIN ^a	ICR In-Service Register
ICRIR ^a	ICR Interrupt-Request Register
ICRIS ^a	ICR Interrupt Status Register
ICRIV	ICR Interrupt Vector Register
ICRL0	ICR Level 0 Control Register (Timer 0)
ICRL2	ICR Level 2 Control Register (DMA 0)
ICRL3	ICR Level 3 Control Register (DMA 1)
ICRL4	ICR Level 4 Control Register (Timer 1)
ICRL5	ICR Level 5 Control Register (Timer 2)
ICRMR ^a	ICR Mask Register
ICRPM ^a	ICR Priority Level Mask Register
ICRSE ^b	ICR Specific EOI Register

^a Register symbol used for both RMX and NON-RMX modes.

^b Write only registers.

^c Read only registers.

OPERATING SYSTEM COMMANDS

Several 8550 and 8540 system operation commands have unique features or display different information when used with the 80186/80188 Emulator. The **memsp** (memory space) system command is not implemented for the 80186/80188 Emulator. Two new emulator specific commands, **spcb** (set PCB) and **lpcb** (locate PCB), are added to the system commands for the 80186/80188 Emulator. The system commands that the 80186/80188 Emulator supports are described in the following pages. For additional information on the 8540/8550 system commands refer to your System Users Manual.

Command Syntax

Each command description includes a syntax block that illustrates the format for a command. This paragraph describes the notation conventions and the format used in the syntax blocks.

Notation Conventions

The syntax block for each command illustrates the command entry: the command name, what parts of the command must be entered, and the order the command parts are entered. Figure 2-26 illustrates a sample syntax block.

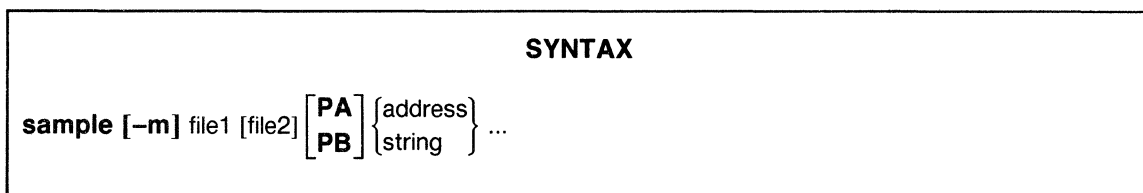


Figure 2-26. Sample syntax block.

Figure 2-26 shows a syntax block for a fictional system command. The command name is **sample**; **-m** is the command modifier; and **file1**, **file2**, **PA**, **PB**, **address**, and **string** are command parameters. The braces, brackets, and trailing dots are for syntactical representation only.

Command Line

A command line begins with a command name and ends with a carriage return. The maximum length of a command line is 80 characters, including spaces and carriage return.

Braces {} in a syntax block surround required parts of the command line. Brackets [] in a syntax block surround optional parts of the command line. When parts are stacked, you choose one part from the stack. Braces and brackets serve only to represent the syntax, and should not be entered as part of the command line.

Boldfaced letters and other characters in the syntax block are required in the command line, and should be entered exactly as they appear in the syntax block.

Three trailing dots in a syntax block show that the preceding elements of the command line may be repeated as many times as needed, up to the maximum line length of 80 characters.

Additional information on the usage of commands and syntax blocks is contained in your System Users Manual.

SYNTAX**al**

or

al {loadaddr} [hiaddr] $\begin{bmatrix} -f \\ -s \end{bmatrix}$ **PARAMETERS**

- loadaddr** Specifies the beginning address of a 4K-byte block of program memory allocated for your program's logical address. If the address is not the beginning address of an even 4K-byte boundary, the memory allocation defaults to the beginning address.
- hiaddr** Specifies the ending address of a 4K-byte block of program memory allocated for your program's logical address. If the address is not the ending address of an even 4K-byte boundary, the memory allocation defaults to the ending address.
- f** Specifies fast memory. No wait states are inserted during program memory accesses. This is the default condition.
- s** Specifies slow memory. Wait states are inserted during program memory accesses.

If you use the **al** command without parameters, the command displays the memory allocation status. If you enter the **al** command with only the **loadaddr** parameter, the command allocates only the 4K-byte block associated with the address.

EXPLANATION

The **al** (allocate) command allocates 4K-byte blocks of program memory for your program's logical addresses.

The **-s** modifier is never required for emulator operation but is available for prototype simulation purposes. The **-s** modifier is used when jumpers P6102 and P7105 (on Emulator Board II) are set to support the modifier. These jumpers are discussed in Section 5 of this manual. Deallocating memory does **not** remove the slow designation: the **-s** modifier remains in effect until changed by a **-f** modifier.

NOTE

*The **al** command does not affect segment register values. Use the **s** (set) command to assign appropriate segment register values to coincide with the allocation.*

EXAMPLES

The following commands allocate memory and display the allocations:

```
> al 0 3fff
  4 BLOCK(S) ALLOCATED   12 BLOCK(S) FREE

> al 8000 -s
  1 BLOCK(S) ALLOCATED   11 BLOCK(S) FREE

> al
00000 - 03FFF
08000 - 08FFF S
  5 BLOCK(S) ALLOCATED   11 BLOCK(S) FREE
```

SYNTAX

$$\mathbf{bk} \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \mathbf{all} \end{array} \right] \mathbf{clr}$$

or

$$\mathbf{bk} [-\mathbf{c}] \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \mathbf{all} \end{array} \right] \text{expression} \left[\begin{array}{c} \mathbf{rd} \\ \mathbf{wt} \end{array} \right] \left[\begin{array}{c} \mathbf{i} \\ \mathbf{m} \end{array} \right] [-\mathbf{a}]$$

PARAMETERS

- 1** Specifies breakpoint 1.
- 2** Specifies breakpoint 2.
- 3** Specifies breakpoint 3.
- all** Specifies all currently defined breakpoints.
- clr** Clears the specified breakpoint.
- c** Continues execution after each breakpoint occurs. If **-c** is not specified (default position), the **bk** command stops execution after a breakpoint occurs. To resume program execution, enter the **g** command without parameters.

expression	An expression representing the address where program execution is interrupted.
rd	Designates that a breakpoint occurs when a memory or I/O read operation occurs at the specified address. Defaults to any access (read or write).
wt	Designates that a breakpoint occurs when a memory or I/O write operation occurs at the specified address. Defaults to any access (read or write).
m	Designates that the breakpoint occurs when a memory read or write occurs at the specified address.
i	Designates that the breakpoint occurs when an I/O read or write occurs at the specified address.
-a	A modifier with which the emulator supports the arming of breakpoints. When breakpoints are armed, the breakpoint conditions must occur in sequence. The actual break and its trace line occur only when the final breakpoint condition is met.

EXPLANATION

The 80186/80188 Emulator allows you to set up to three breakpoints. You can program any of three arming conditions: BK1 to arm BK2, BK2 to arm BK3, or BK1 to arm BK2 to arm BK3. To use this arming feature, first program the first breakpoint, then program the second breakpoint and include the **-a** parameter. To clear the arming feature, you must clear the breakpoint that contains the **-a** parameter.

NOTE

With a breakpoint set and using "trace all" to monitor the execution of the program, the display sometimes shows the displayed breakpoint to be several instructions past the actual breakpoint parameters. This is because the instructions are prefetched before they are executed. The difference in the displayed breakpoint and the actual breakpoint depends on the amount of instructions in the queue.

The 80186/80188 Emulator also allows you to specify read (**rd**), write (**wt**) memory (**m**) and I/O (**i**) operations.

EXAMPLES

The following breakpoints set the emulator to break on any I/O read that follows a memory write to address 1C0F0:

```
> bk 1 1c0f0 m wt
> bk 2 ,,i rd -a
```

Memory Manipulation Commands D, F, LO, MOV, P, and SAV

With the **d** (dump) command, and other commands that manipulate memory (**f**, **lo**, **mov**, **p**, **sav**), the **loadaddr** parameter may be either an absolute address or an address relative to the current Code Segment (CS) Register. The correct syntax for these memory manipulation commands is in your System Users Manual.

EXAMPLES

For example, if CS=0100, the following commands are equivalent:

```
> d 1020
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
001020 12 34 56 78 90 00 00 00 00 00 00 00 00 00 00 00 .4Vx.....

> d CS:20
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000020 12 34 56 78 90 00 00 00 00 00 00 00 00 00 00 00 .4Vx.....
```

NOTE

When you specify an address relative to the current CS, the display includes only the offset value and does not include the CS information.

SYNTAX

```
deal { -a [loadaddr hiaddr] }
```

PARAMETERS

- a** An option that deallocates all program memory blocks.
- loadaddr** An expression representing the beginning address of the deallocated memory block. If no memory space designator is used, all memory spaces in the given range are deallocated.
- hiaddr** An expression representing the ending address of the deallocated memory block. Defaults to the ending address of the 4K-byte block specified by the **loadaddr** parameter.

EXPLANATION

The **deal** (deallocate) command removes the previous allocation of program memory. This command operates on 4K-byte blocks of program memory. The **deal** command does **not** affect previous memory allocations set with the **al** command modifiers **-** and **-s**. If a 4K-byte block is allocated for slow memory, the block can be deallocated with the **deal** command. However, if the block is allocated again, it retains the slow memory allocation, unless changed by the **-f** modifier.

NOTE

*The **deal** command does not affect segment register values. Use the **s** (set) command to assign appropriate segment register values to coincide with the deallocation.*

EXAMPLES

The following are examples of **al** and **deal** commands:

```
> al 0 fff
1 BLOCK(S) ALLOCATED    15 BLOCK(S) FREE

> al 2000 -s
1 BLOCK(S) ALLOCATED    14 BLOCK(S) FREE

> al
00000 - 00FFF
02000 - 02FFF S
2 BLOCK(S) ALLOCATED    14 BLOCK(S) FREE

> deal -a

> al
0 BLOCK(S) ALLOCATED    16 BLOCK(S) FREE

> al 2000
1 BLOCK(S) ALLOCATED    15 BLOCK(S) FREE

> al
02000 - 02FFF S
1 BLOCK(S) ALLOCATED    15 BLOCK(S) FREE
```

SYNTAX

di [loadr] [hiaddr] [lines]

PARAMETERS

- loadr** An expression representing the program/prototype memory address where disassembly begins. Defaults to 0000.
- hiaddr** An expression representing the program/prototype memory address where disassembly ends. Defaults to the end of memory.
- lines** The number of lines to be disassembled. If this parameter and **hiaddr** are omitted, disassembly continues until the end of memory is reached or until you enter CTRL-C.

When you enter the **di** command without parameters, disassembly starts at address 0000 and continues until the end of memory is reached or until you enter CTRL-C.

EXPLANATION

The **di** (disassemble) command translates object code in memory into assembly language instructions. This command displays object code, assembly language mnemonics, and operands.

EXAMPLE

Here is an example of an 80186/80188 Emulator's **di** command output:

LOC	INST	MNEM	OPER
000100	BB0005	MOVW	BX,#0500
000103	B90500	MOVW	CX,#0005
000106	32C0	XORB	AL,AL
000108	0207	ADDB	AL,[BX]
00010A	43	INC	BX
00010B	E2FB	LOOP	\$-03
00010D	BA0710	MOVW	DX, 1007
000110	EE	OUT	DX,AL
000111	90	NOP	
000112	90	NOP	

Diagram illustrating the output format with labels:

- Absolute Address of the Instruction**: Points to the LOC column.
- Hexadecimal Representation of the Instruction**: Points to the INST column.
- Instruction Mnemonic**: Points to the MNEM column.
- Operand(s): the address, register, or data being operated on**: Points to the OPER column.

SYNTAX**ds**

or

ds [-p]**PARAMETERS**

- (default) When the **ds** command is entered without parameters, the short form of the emulator register contents is displayed.
- p** This is a new option for the 80186/80188 Emulator. The **-p** parameter displays the contents of all readable PCB registers. (Some PCB registers are write only and cannot be read.)

EXPLANATION

The **ds** (display status) command entered with no modifier displays the short form of the emulator register contents. The short form display contains the following:

- Status of three hardware inputs to the microprocessor: NMI, INTR, and TEST.
- The frequency of the on-board clock for mode 0 operations or the emulator operating mode (Normal or Queue Status) for modes 1 and 2 operations.
- The contents of the internal general purpose microprocessor registers.
- The bit states of the Flags register.

All numbers in the **ds** command display are hexadecimal. The **-l** (long display modifier) is not supported by the 80186/80188 Emulator.

The **ds -p** command displays the contents of all readable PCB registers. The register contents are displayed in two modes: RMX or NON-RMX. The mode of operation depends on the setting of bit 14 in the Relocation Register (discussed earlier in this section).

EXAMPLES

Here are four examples of the short form display for the **ds** command produced by an 80186/80188 Emulator. Table 2-2 explains the symbols displayed by the short form of the **ds** command.

Mode 0: 8 MHz Clock

> ds

PC/CS:IP	HD STATUS/CLOCK	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F																																		
000000	NMI=1 INTR=1 TEST=1	0000	0000	0000	0000	0000	0000																																		
0000:0000	8MHZ	0000	0000	0000	0000	0000	0000																																		
<table border="0"> <tr> <td>FLAGS</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>OF</td> <td>DF</td> <td>IF</td> <td>TF</td> <td>SF</td> <td>ZF</td> <td>.</td> <td>AF</td> <td>.</td> <td>PF</td> <td>.</td> <td>CF</td> </tr> <tr> <td>0000</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> </tr> </table>								FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF	0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0
FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF																									
0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0																									

Mode 0: 4 MHz Clock

> ds

PC/CS:IP	HD STATUS/CLOCK	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F																																		
000000	NMI=1 INTR=1 TEST=1	0000	0000	0000	0000	0000	0000																																		
0000:0000	4MHZ	0000	0000	0000	0000	0000	0000																																		
<table border="0"> <tr> <td>FLAGS</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>OF</td> <td>DF</td> <td>IF</td> <td>TF</td> <td>SF</td> <td>ZF</td> <td>.</td> <td>AF</td> <td>.</td> <td>PF</td> <td>.</td> <td>CF</td> </tr> <tr> <td>0000</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> </tr> </table>								FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF	0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0
FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF																									
0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0																									

Mode 1 or 2: Queue Status Mode

> ds

PC/CS:IP	HD STATUS/MODE	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F																																		
000000	NMI=1 INTR=1 TEST=1	0000	0000	0000	0000	0000	0000																																		
0000:0000	QUEUE-STATUS MODE	0000	0000	0000	0000	0000	0000																																		
<table border="0"> <tr> <td>FLAGS</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>OF</td> <td>DF</td> <td>IF</td> <td>TF</td> <td>SF</td> <td>ZF</td> <td>.</td> <td>AF</td> <td>.</td> <td>PF</td> <td>.</td> <td>CF</td> </tr> <tr> <td>0000</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> </tr> </table>								FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF	0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0
FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF																									
0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0																									

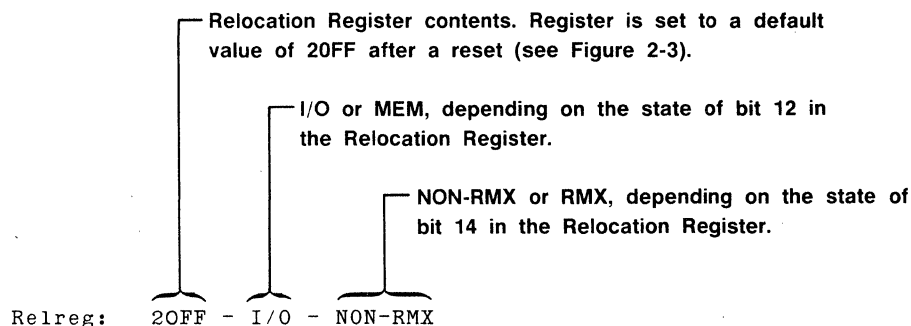
Mode 1 or 2: Normal Mode

> ds

PC/CS:IP	HD STATUS/MODE	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F																																		
000000	NMI=1 INTR=1 TEST=1	0000	0000	0000	0000	0000	0000																																		
0000:0000	NORMAL MODE	0000	0000	0000	0000	0000	0000																																		
<table border="0"> <tr> <td>FLAGS</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>OF</td> <td>DF</td> <td>IF</td> <td>TF</td> <td>SF</td> <td>ZF</td> <td>.</td> <td>AF</td> <td>.</td> <td>PF</td> <td>.</td> <td>CF</td> </tr> <tr> <td>0000</td> <td>X</td> <td>X</td> <td>X</td> <td>X</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> <td>X</td> <td>0</td> </tr> </table>								FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF	0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0
FLAGS	.	.	.	.	OF	DF	IF	TF	SF	ZF	.	AF	.	PF	.	CF																									
0000	X	X	X	X	0	0	0	0	0	0	X	0	X	0	X	0																									

NON-RMX Mode

The NON-RMX mode of operation is the default mode after a reset or when bit 14 in the Relocation Register is set to 0. This is an example of a **ds -p** command display when in NON-RMX mode:



DMA

DMAOCW: XXXX	DMA1CW: XXXX
DMAOTC: XXXX	DMA1TC: XXXX
DMAODP: XXXXX	DMA1DP: XXXXX
DMAOSP: XXXXX	DMA1SP: XXXXX

Chip-Selects

MPCS: XXXX	MMCS: XXXX	PACS: XXXX
LMCS: XXXX	UMCS: XXXX	

Timers

TOMCW: XXXX	T1MCW: XXXX	T2MCW: XXXX
TOMXB: XXXX	T1MXB: XXXX	
TOMXA: XXXX	T1MXA: XXXX	T2MXA: XXXX
TOCNR: XXXX	T1CNR: XXXX	T2CNR: XXXX

Interrupts

ICRIO: XXXX	ICRI1: XXXX	ICRI2: XXXX	ICRI3: XXXX
ICRDO: XXXX	ICRD1: XXXX	ICRTC: XXXX	ICRIS: XXXX
ICRIR: XXXX	ICRIN: XXXX	ICRPM: XXXX	ICRMR: XXXX
ICRPS: XXXX	ICRPR: XXXX	ICRER: XXXX	

See Note

NOTE

The Interrupt Control Register ICRER is a write-only register; the displayed register contents are always "XXXX".

The register symbol names used in the preceding NON-RMX display are defined earlier in Tables 2-16 and 2-17.

The PCB registers are 16-bit registers except for four 20-bit DMA registers (two words). Each 20-bit register is configured from two 16-bit registers (12 bits of each register pair are not used).

The first line in the display shows the Relocation Register's contents. In the preceding display, the "20FF" is the default value of the Relocation Register after a reset. This default value locates the PCB to an address of 0FF00 in I/O space. Figure 2-3 shows the functions of the bits in the Relocation Register.

NOTE

*For an 80188 Emulator, if the user's program modifies the value of the Relocation Register during a **g** (go) command, the address of the PCB is lost until the next reset. When this happens, the error message "Can't find Peripheral Control Block" is displayed. If you know the modified value of the Relocation Register, you can use the **lpcb** command to let the emulating processor know where the PCB was moved.*

RMX Mode

The RMX mode of operation (the 8086/8088 compatibility mode) is set when bit 14 of the Relocation Register is set to 1 (RMX mode). This is an example of a **ds -p** command display when in RMX mode:

Relreg: 50FF - MEM - RMX

DMA

DMAOCW: XXXX	DMA1CW: XXXX
DMAOTC: XXXX	DMA1TC: XXXX
DMAODP: XXXXX	DMA1DP: XXXXX
DMAOSP: XXXXX	DMA1SP: XXXXX

Chip-Selects

MPCS: XXXX	MMCS: XXXX	PACS: XXXX
LMCS: XXXX	UMCS: XXXX	

Timers

TOMCW: XXXX	T1MCW: XXXX	T2MCW: XXXX
TOMXB: XXXX	T1MXB: XXXX	
TOMXA: XXXX	T1MXA: XXXX	T2MXA: XXXX
TOCNR: XXXX	T1CNR: XXXX	T2CNR: XXXX

Interrupts

ICRLO: XXXX	ICRL2: XXXX	ICRL3: XXXX
ICRL4: XXXX	ICRL5: XXXX	ICRIS: XXXX
ICRIR: XXXX	ICRIN: XXXX	ICRPM: XXXX
ICRMR: XXXX	ICRSE: XXXX	ICRIV: XXXX

↑
See Note

NOTE

The Interrupt Control Register ICRSE is a write-only register; the displayed register contents are always "XXXX".

The register symbol names used in the preceding RMX display are defined earlier in Tables 2-16 and 2-17.

SYNTAX
$$\mathbf{g} \begin{bmatrix} -\mathbf{r} \\ -\mathbf{l} \end{bmatrix} [\text{address}]$$
PARAMETERS

- r** Causes the **g** command to be reinvoked each time a breakpoint is encountered. A break message is displayed at each break. This continues indefinitely until you enter CTRL-C.
- l** Same as **-r**, except trace and break lines are suppressed.
- (default) If neither **-r** nor **-l** is specified, the **g** command stops execution after the first break.
- address** An expression representing the value in the Instruction Pointer (IP) Register. The 80186/80188 Emulator uses this information and the information in the Code Segment (CS) Register to calculate the effective starting address (EA).

EXPLANATION

The **g** (go) command begins program execution. The **g** command operates differently for the 80186/80188 Emulator than with other emulators. Usually if you enter the **g** command with an address of 1000, the program counter is set to 1000, and execution begins from that address. With the 80186/80188 Emulator, the **g** command uses information from the Code Segment (CS) Register and Instruction Pointer (IP) to calculate the effective starting address (EA). The EA is calculated as follows:

$$EA = IP + (CS \times 16)$$

IP is the value of the **address** parameter you enter with the **g** command. Because the value of the CS Register may be changed during program execution, you should set the CS Register with the **s** command before running the program. Setting this register ensures that the program starts at the desired address.

NOTE

*The value of the **g** command's **address** parameter must be less than 10000H.*

SYNTAX**lpcb** {address}**PARAMETERS**

address This is a 16-bit address representing the value you believe is in the Relocation Register. This value includes memory or I/O space selection, the interrupt mode, and the PCB's base address. The contents of the Relocation Register are defined earlier in this section in Figure 2-3.

EXPLANATION

The **lpcb** (locate PCB) command is a new system command unique to the 80188 Emulator. When the user's program modifies the value of the Relocation Register during a **g** (go) command, the 80188 Emulator loses the PCB's address. When this happens, the next time the **ds -p** or **spcb** command is used, the error message "Can't find Peripheral Control Block" is displayed. The **lpcb** command permits you to tell the emulator where the PCB was moved. The major difference between the **lpcb** command and the **spcb** command is that the **lpcb** command is used only by the emulating processor. No registers in the PCB are changed in value.

NOTE

If this command is used with the 80186 Emulator, the error message "Command only usable with 80188 Emulator" is displayed.

EXAMPLE

The following is an example of the **lpcb** command for an 8540 or 8550:

```
> lpcb 60ff
```

This 16-bit address is the value you believe is in the Relocation Register.

The following is an example of the **lpcb** command for an 8560 or other host in term mode:

```
> 8540 lpcb 60ff
```

SYNTAX

map option {loadaddr [hiaddr]} ...

PARAMETERS

option	A list of effective options is contained in your System Users Manual.
loadaddr	An expression representing the lower bound of the address range assigned to program or user prototype memory. The lower bound of the address range starts at the specified address rounded down to a 4K-byte multiple.
hiaddr	An expression representing the upper bound of the address range assigned to program or user prototype memory. The upper bound of the address range starts at the specified address rounded up to a 4K-byte multiple. The hiaddr parameter must be greater than or equal to loadaddr parameter. This parameter defaults to the end of the 4K-byte block that contains the lower address.

When you enter the **map** command without parameters, the current memory map assignments are displayed in tabular form.

EXPLANATION

The 80186/80188 Emulator supports the **map** command described in your System Users Manual, with the following exceptions:

- The 80186/80188 Emulator maps memory in 4K-byte blocks.
- The **-m** modifier is not supported.
- Memory segment information may not include in the **map** command.

SYNTAX**mem** [loadaddr [hiaddr]]**PARAMETERS**

loadaddr An expression representing the beginning of a block of memory the emulator is allowed to access.

hiaddr An expression representing the end of a block of memory the emulator is allowed to access.

When you enter the **mem** command with out parameters, the current memory status is displayed.

EXPLANATION

The **mem** (memory) command informs the emulator that the prototype contains memory at a given block of addresses. This command operates on 4K-byte blocks. The **mem** command can also be used to reverse a previous **nomem** (no memory) command. The default condition is that all prototype memory is available to the program. Memory segment information is not valid with the **mem** command. For an example of **mem** command usage, see the discussion of the **nomem** command.

MEMSP—Defines memory space

The **memsp** (memory space) command is not implemented for the 80186/80188 Emulator.

SYNTAX**nomem** [loadr [hiaddr]]**PARAMETERS**

- loadr** An expression representing the beginning of a block of memory the emulator is not allowed to access. May include memory space designators.
- hiaddr** An expression representing the end of a block of memory the emulator is not allowed to access.

Using the **nomem** command with out parameters displays the list of memory blocks which are currently unavailable. Each memory space has a separate display.

EXPLANATION

The **nomem** (no memory) command informs the emulator that the prototype does **not** contain memory at a given block of addresses. The default condition is that all prototype memory is available to the program. The **nomem** command operates on 4K-byte blocks. Memory segment information is not valid with the **nomem** command.

The **mem** command reverses the effect of the **nomem** command.

EXAMPLE

The following are examples of **mem** and **nomem** commands:

```
> nomem 0 7ffff

> nomem
INVALID USER MEMORY
00000 - 7FFFF

> mem 0

> mem
VALID USER MEMORY
00000 - 00FFF
80000 - FFFFF

> nomem
INVALID USER MEMORY
01000 - 7FFFF
```

SYNTAX

```
rd [-m] [  $\begin{bmatrix} -b \\ -w \end{bmatrix}$  ] portnum ...
```

PARAMETERS

- m** Specifies that the value of portnum is a memory address (valid for memory-mapped I/O).
If this modifier is omitted, the **portnum** parameter defaults to a fixed port read.
- b** Specifies byte-oriented reading. Default value.
- w** Specifies word-oriented reading.
- portnum** An expression representing an I/O port. If **-m** is not used, the expression designates a fixed I/O port. If **-m** is used, the expression designates a memory location (memory-mapped I/O). The expression may include only one memory space designator. You may include symbolic names for the expression.

EXPLANATION

The **rd** (read) command reads a byte or a word from a prototype I/O port. If more than one **portnum** value is entered, reads are performed and the results displayed in the order the **portnum** parameters are entered.

NOTE

*Do **not** use the **rd** command to read from the area where the PCB is located. The **ds -p** command should be used to display the contents of the PCB registers.*

The 80186/80188 Emulator supports both fixed-port and memory-mapped I/O. The default is fixed-port I/O. The I/O ports must be in the range 0000-FFFF. The 80186/80188 Emulator does **not** support the **-s** modifier of the **rd** command.

NOTE

*The **rd** command **always** reads from the prototype. In mode 0, the command internally changes to mode 1 to execute the command, and then returns to mode 0. An error message occurs if the probe is not connected to the prototype when this command is executed.*

SYNTAX

reset

EXPLANATION

The **reset** command sends a hardware reset signal to the emulating microprocessor and clears any pending interrupts. The "Value After Reset" column of Table 2-2 indicates which 80186/80188 Emulator registers are affected by the **reset** command.

EXAMPLE

Suppose the **ds** command returns the following status:

```
> ds
PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000106        NMI=0 INTR=0 TEST=1  000F 0002 0000 0C30 1034 0100
0000:0106     5MHZ                0502 1007 1223 0000 0FF0 F006

  FLAGS      . . . .  OF DF IF TF  SF ZF  . AF  . PF  . CF
FO06 X X X X    0 0 0 0    0 0 X 0    X 1 X 0
```

Enter the **reset** command, then use the **ds** command to check the results. The arrows show the changed registers.

```
> reset
> ds
↓ ↓
PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000000        NMI=0 INTR=0 TEST=1  000F 0002 0000 0C30 0000 0000
FFFF:0000     5MHZ                0502 1007 1223 0000 0000 0000

  FLAGS      . . . .  OF DF IF TF  SF ZF  . AF  . PF  . CF
0000 X X X X    0 0 0 0    0 0 X 0    X 0 X 0
↑ ↑ ↑ ↑      ↑ ↑ ↑ ↑      ↑ ↑      ↑      ↑      ↑
```

SYNTAX

s symbolspec=expression ...

PARAMETERS

symbolspec	The name of the symbol or register to be changed. The symbolspec parameter may be a standard register name or a user-created symbol that is already in the symbol table.
expression	Any valid expression as defined under “Legal Address Expressions” in your System Users Manual.

EXPLANATION

The **s** (set) command changes the values of the 80186 or 80188 Emulator registers. The registers' symbols and the registers these symbols represent are shown in Table 2-2.

You may use more than one symbolspec and expression in a command line.

NOTE

*Do **not** use the **s** command to set the registers in the PCB. The **spcb** command should be used to set or change the values in the PCB's registers.*

SYNTAX

sel [chip]

PARAMETER

chip	The name of the target processor.
------	-----------------------------------

When you enter the **sel** command without parameters, the currently selected processor name is displayed.

EXPLANATION

The **sel** (select) command permits you to select the 80186 or 80188 Emulator depending on which prototype control probe you have attached to the 80186/80188 Emulator boards.

EXAMPLES

For an 8550, enter “80186” or “80188” to select either an 80186 or 80188 Emulator. The 8550 **sel** command for an 80186 Emulator looks like this:

```
> sel 80186
80186 Emulator V n.nn mm/dd/yy
```

Diagram illustrating the components of the command output:

- 80186 Emulator**: Name of Emulator
- V n.nn**: Software Version Number
- mm/dd/yy**: Month/Day/Year of Software Version

For an 8540, always select the 80186. The system will return the microprocessor name that matches the prototype control probe installed. The 8540 **sel** command for an 80188 Emulator looks like this:

```
> sel 80186
80188 Emulator V n.nn mm/dd/yy
```

Selecting an Assembler

To select the 80186/80188 assembler in the 8560, enter:

```
$ uP=80186; export uP
```

NOTE

The 8550 development system does not have an 80186/80188 Assembler. If you have an 8086/8088 Assembler for the 8550, that assembler can be used to assemble and load a program that does not contain the additional 80186/80188 instructions unavailable to the 8086/8088 microprocessor.

To select the 8086/8088 Assembler in the 8550, enter:

```
> sel 8086
8086 Emulator V n.nn mm/dd/yy
```

After the program has assembled and before the program is loaded into program memory, enter the **sel** command again with either the “80186” or “80188” parameter to select the 80186 or 80188 Emulator.

SYNTAX

spcb symbolspec=value ...

PARAMETERS

symbolspec	The name of the register in the PCB that is to be changed. Refer to Tables 2-16 and 2-17 for a list of PCB register names.
value	The 16-bit value of the register.

EXPLANATION

The **spcb** (set PCB) command is a new system command unique to the 80186/80188 Emulator. The **spcb** command writes to the registers within the PCB. The **ds -p** (display status) command displays the contents of all the PCB registers. When a value is entered for a particular PCB register, the value is not checked before sending the value to the peripheral register. Refer to Tables 2-16 and 2-17 for a list of the registers and register symbol names. Symbol names may be entered in lowercase or uppercase.

The relocation address for the PCB is contained in the lower 12 bits of the PCB's Relocation Register (refer to Figure 2-3). The PCB's relocation address should not be set in the lowest 1000 of I/O address space or in the same I/O area that contains the SVC ports. The PCB's relocation address also should not be mapped into the lowest 8K bytes of memory (0000-2FFF). None of these conditions are reported as an error.

You must be careful where you locate the emulating processor's stack pointer (SP) in relationship to the PCB's base address. When a break is generated, the emulating processor pushes its registers onto the stack, causing six additional stack writes to take place from the SP's address. (These stack pushes are not seen by the prototype circuitry.) If the SP's address is within six locations of the address area reserved by the PCB (256 bytes), the area is altered and could cause the emulating processor to crash. For example, when the PCB's base address is set for 0FF00 in memory space, an address range of 0FF00-0FFFF is the area reserved for the PCB. If the SP is set within the address range of 00000-00005, the next system break causes the emulating processor to push its registers into the area reserved for the PCB. In this example, the SP must be set for an address of 00006 or higher to prevent altering of the PCB's area.

EXAMPLES

To change the value of the Relocation Register for an 8540 or 8550, enter:

```
> spcb RELREG=30FF
```

To change the value of the Relocation Register for an 8560 or other host in term mode, enter:

```
> 8540 spcb relreg=30ff
```

You cannot write to two of the ICRs (Interrupt Control Registers) within the PCB. These registers are the ICRPR (ICR Poll Register) and the ICRPS (ICR Poll Status Register). If you attempt to write to these registers, an error message "Register not writable:" is displayed.

NOTE

*You should use caution when writing to the registers in the PCB. The register symbols and values (entered with the **spcb** command) are not checked before writing to the designated register. A check is made to see if the designated register is available and that the PCB address can be read. The accuracy of the register values is not checked. Because various types of registers exist (especially the timer registers), the values written to the registers with the **spcb** command may not equal the register values displayed with the **ds -p** command.*

NOTE

*For an 80188 Emulator, if the user's program modifies the value of the Relocation Register during a **g** (go) command, the address of the PCB is lost until the next reset. When this happens, an error message "Can't find Peripheral Control Block" is displayed the next time either the **ds -p** or **spcb** command is used. If you know the modified value of the Relocation Register, the **lpcb** command can be used to let the emulating processor know where the PCB was moved.*

SYNTAX

tra [-n]

or

tra [-s] [-n] $\left\{ \begin{array}{l} \text{all} \\ \text{jmp} \\ \text{off} \end{array} \right\}$ [loadaddr] [hiaddr]

PARAMETERS

- s** Stops program execution after each trace line is displayed. If **-s** is not specified, the program continues execution after each trace line is displayed.
- n** Selects normal display. The contents of only the most important registers are displayed.
- all** Every instruction is displayed after its execution.
- jmp** Displays only jump instructions: unconditional jumps, subroutine calls, and conditional branches where the specified condition is satisfied.
- off** Disables trace display.
- loadaddr** An expression representing the lower bound of the address range to which **all**, **jmp**, or **off** applies. Defaults to 0.
- hiaddr** An expression representing the upper bound of the address range to which **all**, **jmp**, or **off** applies. Defaults to the top of memory. The **hiaddr** parameter must be greater than **loadaddr** parameter. If both **loadaddr** and **hiaddr** are specified and **loadaddr** does not contain the first byte of an instruction, tracing starts at the next instruction after **loadaddr**.

When you enter the **tra** command without parameters, the current trace conditions are displayed.

EXPLANATION

The **tra** (trace) command selects the range and type of instructions the system displays as your program executes. The short form of the **tra** command displays register values for the 80186/80188 microprocessors.

NOTE

*Segment registers are not allowed in the trace address parameters because these registers may be changed during program execution. If you enter a segment register as part of a trace parameter, an error will occur when you enter the **g** (go) command.*

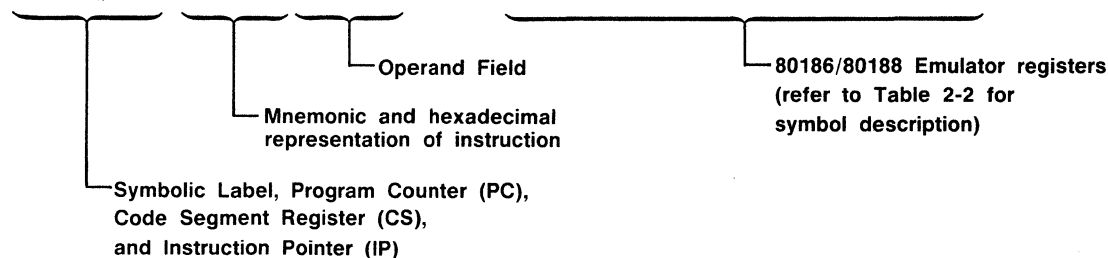
EXAMPLES

Here is a sample 80186/80188 trace display with symbolic debug on:

> tra all

> g START

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX,#0500	0063	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEMO+000103							
000103	MOVW CX,#0005	0063	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
DEMO+000106							
000106	XORB AL,AL	0000	0005	0000	0000	0000	0000
0000:0108	32C0	0500	1007	0000	0000	0000	F046
AL00P							
000108	ADDB AL,[BX]	0000	0005	0000	0000	0000	0000
0000:010A	0207	0500	1007	0000	0000	0000	F046
DEMO+00010A							
00010A	INC BX	0000	0005	0000	0000	0000	0000
0000:010B	43	0501	1007	0000	0000	0000	F002



NOTE

In the preceding trace display, PC is the **last** instruction executed while CS:IP is the **next** instruction executed. When using the **ds** command with no modifier, the short form display shows both PC and CS:IP as the next instruction executed.

Here is a sample **80186 Emulator** trace display of a program. This program changes the PCB's base address, sets the emulator for a DMA transfer of 34 bytes, and breaks program execution after the transfer is completed.

```
> tra all
> g 0
```

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
000000	MOVW AX,#3080	3080	0000	0000	0000	0000	0000
0000:0003	B88030	0000	FFFE	0000	0000	0000	F002
000003	MOVW DX,#FFFE	3080	0000	0000	0000	0000	0000
0000:0006	BAFEFF	0000	FFFE	0000	0000	0000	F002
000006	OUT DX,AX	3080	0000	0000	0000	0000	0000
0000:0007	EF	0000	FFFE	0000	0000	0000	F002
000007	MOVW AX,80FE	3080	0000	0000	0000	0000	0000
0000:000A	A1FE80	0000	FFFE	0000	0000	0000	F002
00000A	MOVW 0300,AX	3080	0000	0000	0000	0000	0000
0000:000D	A30003	0000	FFFE	0000	0000	0000	F002
00000D	MOVW AX,#0100	0100	0000	0000	0000	0000	0000
0000:0010	B80001	0000	FFFE	0000	0000	0000	F002
000010	MOVW 80C0,AX	0100	0000	0000	0000	0000	0000
0000:0013	A3C080	0000	FFFE	0000	0000	0000	F002
000013	MOVW AX,#0000	0000	0000	0000	0000	0000	0000
0000:0016	B80000	0000	FFFE	0000	0000	0000	F002

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
000016	MOVW 80C2,AX	0000	0000	0000	0000	0000	0000
0000:0019	A3C280	0000	FFFE	0000	0000	0000	F002
000019	MOVW AX,#0200	0200	0000	0000	0000	0000	0000
0000:001C	B80002	0000	FFFE	0000	0000	0000	F002
00001C	MOVW 80C4,AX	0200	0000	0000	0000	0000	0000
0000:001F	A3C480	0000	FFFE	0000	0000	0000	F002
00001F	MOVW AX,#0000	0000	0000	0000	0000	0000	0000
0000:0022	B80000	0000	FFFE	0000	0000	0000	F002
000022	MOVW 80C6,AX	0000	0000	0000	0000	0000	0000
0000:0025	A3C680	0000	FFFE	0000	0000	0000	F002
000025	MOVW AX,#0034	0034	0000	0000	0000	0000	0000
0000:0028	B83400	0000	FFFE	0000	0000	0000	F002

```

000028      MOVW      80C8,AX      0034 0000 0000 0000 0000 0000
0000:002B      A3C880      0000 FFFE 0000 0000 0000 F002

```

```

00002B      MOVW      AX,#B70F      B70F 0000 0000 0000 0000 0000
0000:002E      B80FB7      0000 FFFE 0000 0000 0000 F002

```

```

PC/CS:IP  MNEMONIC/DATA      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
00002E      MOVW      80CA,AX      B70F 0000 0000 0000 0000 0000
0000:0031      A3CA80      0000 FFFE 0000 0000 0000 F002

```

Internal	DMAOCW	DMAOTC	DMAODP	DMAOSP	DMA1CW	DMA1TC	DMA1DP	DMA1SP
DMA	B70B	0032	00204	00104	0000	2DFD	09A66	0E9FD
DMA	B70B	0030	00208	00108	0000	2DFD	09A66	0E9FD
DMA	B70B	002E	0020C	0010C	0000	2DFD	09A66	0E9FD
DMA	B70B	002C	00210	00110	0000	2DFD	09A66	0E9FD
DMA	B70B	002A	00214	00114	0000	2DFD	09A66	0E9FD
DMA	B70B	0028	00218	00118	0000	2DFD	09A66	0E9FD
DMA	B70B	0026	0021C	0011C	0000	2DFD	09A66	0E9FD
DMA	B70B	0024	00220	00120	0000	2DFD	09A66	0E9FD
DMA	B70B	0022	00224	00124	0000	2DFD	09A66	0E9FD
DMA	B70B	0020	00228	00128	0000	2DFD	09A66	0E9FD
DMA	B70B	001E	0022C	0012C	0000	2DFD	09A66	0E9FD
DMA	B70B	001C	00230	00130	0000	2DFD	09A66	0E9FD
DMA	B70B	001A	00234	00134	0000	2DFD	09A66	0E9FD
DMA	B70B	0018	00238	00138	0000	2DFD	09A66	0E9FD
DMA	B70B	0016	0023C	0013C	0000	2DFD	09A66	0E9FD
DMA	B70B	0014	00240	00140	0000	2DFD	09A66	0E9FD
DMA	B70B	0012	00244	00144	0000	2DFD	09A66	0E9FD
DMA	B70B	0010	00248	00148	0000	2DFD	09A66	0E9FD
DMA	B70B	000E	0024C	0014C	0000	2DFD	09A66	0E9FD
DMA	B70B	000C	00250	00150	0000	2DFD	09A66	0E9FD
DMA	B70B	000A	00254	00154	0000	2DFD	09A66	0E9FD

Internal	DMAOCW	DMAOTC	DMAODP	DMAOSP	DMA1CW	DMA1TC	DMA1DP	DMA1SP
DMA	B70B	0008	00258	00158	0000	2DFD	09A66	0E9FD
DMA	B70B	0006	0025C	0015C	0000	2DFD	09A66	0E9FD
DMA	B70B	0004	00260	00160	0000	2DFD	09A66	0E9FD
DMA	B70B	0002	00264	00164	0000	2DFD	09A66	0E9FD
DMA	B709	0000	00268	00168	0000	2DFD	09A66	0E9FD

```

PC/CS:IP  MNEMONIC/DATA      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000031      JMP      $-0000      B70F 0000 0000 0000 0000 0000
0000:0031      E9FDFF      0000 FFFE 000Q 0000 0000 F002

```

```

000031  <BREAK      TRACE, BKPT1>

```

This display for an 80186 Emulator shows the execution of each instruction before and after the DMA transfer. The **tra all** command displays the contents of the emulating processor's general registers for each instruction executed. The DMA register symbol names at the beginning of each column are defined in Table 2-3. During a DMA transfer the microprocessor's general registers remain unchanged and would have no meaning if they were displayed. Therefore, during the DMA transfer, the contents of the internal DMA registers are displayed to show the internal DMA activity. During the DMA transfer, one transferred byte out of every two bytes is displayed. In the preceding display during the DMA transfer, the DMA registers for both DMA channels are shown, however, only channel 0 shows the DMA activity.

When executing a program using the **80188 Emulator**, if the program modifies the value of the PCB's base address (by changing the Relocation Register contents) the emulator loses the PCB's base address. The **lpcb** command is used to tell the emulator where the PCB was moved. The following is a sample 80188 Emulator's trace display of the previous program. The program changed the PCB's base address, set the emulating processor for a DMA transfer of 34 bytes, and broke program execution after the transfer was completed.

```
> tra all
> g 0
```

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
000000	MOVW AX,#3080	3080	0000	0000	0000	0000	0000
0000:0003	B88030	0000	FFFE	0000	0000	0000	F002
000003	MOVW DX,#FFFE	3080	0000	0000	0000	0000	0000
0000:0006	BAFEFF	0000	FFFE	0000	0000	0000	F002
000006	OUT DX,AX	3080	0000	0000	0000	0000	0000
0000:0007	EF	0000	FFFE	0000	0000	0000	F002
000007	MOVW AX,80FE	3080	0000	0000	0000	0000	0000
0000:000A	A1FE80	0000	FFFE	0000	0000	0000	F002
00000A	MOVW 0300,AX	3080	0000	0000	0000	0000	0000
0000:000D	A30003	0000	FFFE	0000	0000	0000	F002
00000D	MOVW AX,#0100	0100	0000	0000	0000	0000	0000
0000:0010	B80001	0000	FFFE	0000	0000	0000	F002
000010	MOVW 80C0,AX	0100	0000	0000	0000	0000	0000
0000:0013	A3C080	0000	FFFE	0000	0000	0000	F002
000013	MOVW AX,#0000	0000	0000	0000	0000	0000	0000
0000:0016	B80000	0000	FFFE	0000	0000	0000	F002

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
000016	MOVW 80C2,AX	0000	0000	0000	0000	0000	0000
0000:0019	A3C280	0000	FFFE	0000	0000	0000	F002
000019	MOVW AX,#0200	0200	0000	0000	0000	0000	0000
0000:001C	B80002	0000	FFFE	0000	0000	0000	F002
00001C	MOVW 80C4,AX	0200	0000	0000	0000	0000	0000
0000:001F	A3C480	0000	FFFE	0000	0000	0000	F002
00001F	MOVW AX,#0000	0000	0000	0000	0000	0000	0000
0000:0022	B80000	0000	FFFE	0000	0000	0000	F002
000022	MOVW 80C6,AX	0000	0000	0000	0000	0000	0000
0000:0025	A3C680	0000	FFFE	0000	0000	0000	F002
000025	MOVW AX,#0034	0034	0000	0000	0000	0000	0000
0000:0028	B83400	0000	FFFE	0000	0000	0000	F002
000028	MOVW 80C8,AX	0034	0000	0000	0000	0000	0000
0000:002B	A3C880	0000	FFFE	0000	0000	0000	F002
00002B	MOVW AX,#B70F	B70F	0000	0000	0000	0000	0000
0000:002E	B80FB7	0000	FFFE	0000	0000	0000	F002

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
00002E	MOVW 80CA,AX	B70F	0000	0000	0000	0000	0000
0000:0031	A3CA80	0000	FFFE	0000	0000	0000	F002

No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 No DMA Trace: PCB Location lost. Use LPCB cmd.
 00002E <BREAK ESC, TRACE>

This display for an 80188 Emulator shows the execution of each instruction prior to the DMA transfer. The emulator loses the location of the PCB's base address after the first instruction is executed. When the DMA transfer starts the emulator doesn't know where the PCB is located. Therefore, the emulator cannot display the contents of the DMA registers. Instead the emulator displays the message "No DMA Trace: PCB Location Lost. Use LPCB cmd." This message is displayed once for every two data bytes transferred. When this message is displayed, you can stop program execution by pressing the CTRL-C key. You can then enter the new PCB's location with the **lpcb** command and continue execution. The DMA transfer continues this time displaying the DMA registers and showing the internal DMA activity.

```
> lpcb 3080
```

```
> g
```

Internal	DMAOCW	DMAOTC	DMAODP	DMAOSP	DMA1CW	DMA1TC	DMA1DP	DMA1SP
DMA	B70B	0020	00214	00114	0000	FEF7	0FCDO	0A5CD
DMA	B70B	001E	00216	00116	0000	FEF7	0FCDO	0A5CD
DMA	B70B	001C	00218	00118	0000	FEF7	0FCDO	0A5CD
DMA	B70B	001A	0021A	0011A	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0018	0021C	0011C	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0016	0021E	0011E	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0014	00220	00120	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0012	00222	00122	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0010	00224	00124	0000	FEF7	0FCDO	0A5CD
DMA	B70B	000E	00226	00126	0000	FEF7	0FCDO	0A5CD
DMA	B70B	000C	00228	00128	0000	FEF7	0FCDO	0A5CD
DMA	B70B	000A	0022A	0012A	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0008	0022C	0012C	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0006	0022E	0012E	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0004	00230	00130	0000	FEF7	0FCDO	0A5CD
DMA	B70B	0002	00232	00132	0000	FEF7	0FCDO	0A5CD
DMA	B709	0000	00234	00134	0000	FEF7	0FCDO	0A5CD

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
000031	JMP \$-0000	B70F	0000	0000	0000	0000	0000
0000:0031	E9FDFF	0000	FFFE	0000	0000	0000	F002

```
000031 <BREAK TRACE, BKPT1>
```

SYNTAX

```
wrt [-m]  $\begin{bmatrix} -b \\ -w \end{bmatrix}$  portnum value
```

PARAMETERS

-m	Specifies that the value of portnum is a memory address (valid for memory-mapped I/O). If this modifier is omitted, the portnum parameter defaults to a fixed port write.
-b	Specifies byte-oriented writing. Default value.
-w	Specifies word-oriented writing.
portnum	An expression representing an I/O port. If -m is not used, the expression designates a fixed I/O port. If -m is used, the expression designates a memory location (memory-mapped I/O). The expression may include only one memory space designator. You may include symbolic names for the expression.
value	Any valid expression used to show the data byte or word written to the I/O port. The value parameter must not exceed the maximum possible value of the indicated word or byte.

EXPLANATION

NOTE

*Do **not** use the **wrt** command to write to the area where the PCB is located. Only the **spcb** command should be used to change the contents of the PCB registers.*

The **wrt** (write) command writes a byte or a word to an I/O port. The 80186/80188 Emulator supports both fixed-port and memory-mapped I/O. The default condition is fixed-port I/O. The I/O ports must be in the range 0000-FFFF. The 80186/80188 Emulator does not support the **-s** modifier of the **wrt** command.

NOTE

*The **wrt** command **always** writes to the prototype. In mode 0, the command internally changes to mode 1 to execute the command and then returns to mode 0. An error message occurs if the probe is not connected to the prototype when you enter this command.*

X—Loads and executes program

The **x** (execute) command loads a file and executes a **g** (go) command at the transfer address. On the 80186/80188 Emulator, the **x** command is supported **only** under the following conditions:

- The CS Register is set to 0.
- The transfer address is within the range 0000–FFFF.

Since the transfer address is an absolute address containing both CS (Code Segment) and IP (Instruction Pointer) information, you will rarely use the **x** command with the 80186/80188 Emulator.

Refer to “Special Considerations” later in this section for more information on program execution.

ERROR MESSAGES

Two error messages are associated with the **spcb** command.

Register not compatible with RMX mode: This error message is displayed when an Interrupt Register name assignment is made and the register name is not valid in the current interrupt mode.

Register not writable: This error message is displayed when you attempt to write to either the ICRPR (Interrupt Control Registers—Poll Register) or the ICRPS (Interrupt Control Registers—Poll Status Register). Table 2-17 shows the read only registers in the Interrupt Controller.

Two error messages are associated with the **lpcb** command.

Can't find Peripheral Control Block: This error message is displayed for an 80188 Emulator only. If the user's program modifies the value of the Relocation Register during a **g** (go) command, the address of the PCB is lost until the next reset. When this happens, the next time the **ds -p** or **spcb** command is used this error message is displayed. If you know the modified value of the Relocation Register, the **lpcb** command can be used to let the emulating processor know where the PCB was moved.

Command only usable with 80188 Emulator: This error message is displayed when the **lpcb** command is used with an 80186 Emulator.

REAL-TIME PROTOTYPE ANALYZER (RTPA)

The 80186/80188 Emulator supports the Trigger Trace Analyzer (TTA) rather than the Real-Time Prototype Analyzer (RTPA).

TRIGGER TRACE ANALYZER (TTA) COMMANDS

The following text discusses the emulator-specific commands of the Trigger Trace Analyzer (TTA) option. For more information about the TTA, refer to the *Trigger Trace Analyzer Users Manual*, or to the TTA discussion in the emulation section of your System Users Manual.

TTA Bus Operation Designators—BUS and EVE

Table 2-18 lists the 80186/80188 bus operation designators recognized by the **bus** command and by the **-b** parameter of the **eve** command for the TTA.

Table 2-18
80186/80188 TTA Bus Operation Designators

Symbol	Bus Operation Type
BHE	Bus high enable (80186)
CLR	All types
DMA	DMA cycle
F	First fetch
FS	Subsequent fetches
IA	Interrupt acknowledge
INT	Interrupt request
IR	I/O read
IW	I/O write
LCK	Bus priority lock control
RD	Memory reads
TST	Test control for WAIT instruction
WT	Memory writes

Set Consecutive Events—CONS

The 80186/80188 Emulator supports the **cons** command CYC mode of operation. The 80186/80188 Emulator does **not** support the **cons** command FET mode or EMU mode of operation.

Display Contents of TTA's Acquisition Memory—DISP

The **disp** command displays the contents of the TTA's Acquisition Memory.

NOTE

When breakpoints are set, the display sometimes shows that the displayed breakpoint is several instructions past the actual breakpoint parameters. This occurs because the instructions are prefetched before they are executed. The difference in the displayed breakpoint and the actual breakpoint depends on the amount of instructions in the queue.

Here is an example of the **disp** command for an 80186/80188 Emulator:

```
> disp
ADDR  DATA  MNEM   OPER          7-PROBE-O  BUS          LCK
000100 BB    MOVW    BX,#0500  1111 1111    F  BHE TST
000101 00                                1111 1111    FS BHE TST
000102 05                                1111 1111    FS BHE TST
000103 B9    MOVW    CX,#0005  1111 1111    F  BHE TST
000104 05                                1111 1111    FS BHE TST
000105 00                                1111 1111    FS BHE TST
000106 32    XORB    AL,AL     1111 1111    F  BHE TST
000107 C0                                1111 1111    FS BHE TST
000108 02    ADDB    AL,[BX]   1111 1111    F  BHE TST
000109 07                                1111 1111    FS BHE TST
000500 00                                1111 1111    RD   TST
00010A 43    INC     BX        1111 1111    F  BHE TST
00010B E2    LOOP    $-03      1111 1111    F  BHE TST
00010C FB                                1111 1111    FS BHE TST
000108 02    ADDB    AL,[BX]   1111 1111    F  BHE TST
000109 07                                1111 1111    FS BHE TST
000501 00                                1111 1111    RD BHE TST
00010A 43    INC     BX        1111 1111    F  BHE TST
00010B E2    LOOP    $-03      1111 1111    F  BHE TST
00010C FB                                1111 1111    FS BHE TST
000108 02    ADDB    AL,[BX]   1111 1111    F  BHE TST
000109 07                                1111 1111    FS BHE TST
```

The following display is an example of the **disp** command's output with both **tra all** and symbolic debug on. The information at addresses 8, 9, and A appears each time the operating system generates a break. (The system breaks at every instruction during **tra all**.) The LCK bus transaction occasionally appears when the system resumes execution and does not signify an error.

```
> disp
ADDR  DATA MNEM      OPER          7-PROBE-O  BUS

START
000100 BB  MOVW      BX,#0500      1111 1111  F BHE TST      LCK
000101 00                                1111 1111  FS BHE TST
000102 05                                1111 1111  FS BHE TST
000008 7F                                1111 1111  RD BHE TST
000009 EF                                1111 1111  RD BHE TST
00000A 7F                                1111 1111  RD BHE TST

DEMO+000103
000103 B9  MOVW      CX,#0005      1111 1111  F BHE TST
000104 05                                1111 1111  FS BHE TST
000105 00                                1111 1111  FS BHE TST
000008 7F                                1111 1111  RD BHE TST
000009 77                                1111 1111  RD BHE TST
00000A 7F                                1111 1111  RD BHE TST

DEMO+000106
000106 32  XORB      AL,AL          1111 1111  F BHE TST      LCK
000107 C0                                1111 1111  FS BHE TST
000008 7F                                1111 1111  RD BHE TST
000009 E7                                1111 1111  RD BHE TST
```

The following display is an example of the **disp** command showing DMA activity under the bus operation designators:

```
> disp
ADDR  DATA MNEM      OPER          7-PROBE-O  BUS

00002C OF                                1111 1111  FS BHE TST
00002D B7                                1111 1111  FS BHE TST
00002E A3  MOVW      80CA,AX        1111 1111  F BHE TST
00002F CA                                1111 1111  FS BHE TST
000030 80                                1111 1111  FS BHE TST
0080CA OF                                1111 1111  WT BHE TST
0080CB B7                                1111 1111  WT BHE TST
000100 90                                1111 1111  RD BHE TST DMA
000101 90                                1111 1111  RD BHE TST DMA
000031 E9  JMP      $-0000          1111 1111  F BHE TST
000200 90                                1111 1111  WT BHE TST DMA
000201 90                                1111 1111  WT BHE TST DMA
000032 FD                                1111 1111  FS BHE TST
000102 90                                1111 1111  RD BHE TST DMA
000103 90                                1111 1111  RD BHE TST DMA
000033 FF                                1111 1111  FS BHE TST
000202 90                                1111 1111  WT BHE TST DMA
000203 90                                1111 1111  WT BHE TST DMA
```

The following display is an example of the **disp** command showing interrupts and locked interrupt instructions under the bus operation designators:

```
> disp
ADDR  DATA  MNEM      OPER      7-PROBE-O  BUS
000000 FO    LOCK SBBB   AL,[BX][SI] 1111 1111  F BHE      INT
000001 1A                      1111 1111  FS BHE      INT
000002 00                      1111 1111  FS BHE      INT
00043C 00                      1111 1111  RD          INT LCK
000003 FO    LOCK MOVW   AX,0000      1111 1111  F BHE      INT
000004 A1                      1111 1111  FS BHE      INT
000005 00                      1111 1111  FS BHE      INT
000006 00                      1111 1111  FS BHE      INT
000000 FO                      1111 1111  RD BHE      INT LCK
000001 1A                      1111 1111  RD BHE      INT LCK
000007 00    ADDB       [BX],CL        1111 1111  F BHE      INT
000008 0F                      1111 1111  FS BHE      INT
00043C 00                      1111 1111  RD          INT
00043C A8                      1111 1111  WT          INT
000009 E1    LOOPZ     $+02            1111 1111  F BHE      INT
00000A 00                      1111 1111  FS BHE      INT
```

SPECIAL CONSIDERATIONS

The following paragraphs describe the special considerations unique to the 80186/80188 Emulator.

The HALT Instruction

In all three emulation modes, a HALT instruction causes the emulator to halt. Control does **not** automatically return to the operating system. Press CTRL-C to return control to the operating system.

Emulator Fault Error

When you select the 80186/80188 Emulator, a section of system code is loaded in the emulator. This code is checked each time you start the emulator. If this code is corrupted in some way, the development system's error "45—Emulator faulted" is displayed, and you must reselect the emulator.

The emulator faulted error can also occur if you lose the clock in emulation modes 1 or 2. You will lose the clock if you turn off your prototype while the development system is in emulation mode 1 or 2. You can avoid this error by changing to emulation mode 0 before turning off your prototype.

Interrupt Status Register Constraints

During code execution, DMA activity can be prevented by setting bit 15 in the Interrupt Status Register of the PCB. Setting bit 15 (DHLT-DMA halt transfer bit) high disables all DMA activity. Bit 15 is automatically set high whenever an NMI occurs. Bit 15 is cleared by an interrupt return (IRET) instruction or a system reset.

When breaking or escaping from user code execution, the 80186/80188 Emulator uses an NMI. The NMI sets bit 15 in the Interrupt Status Register high. When the emulator returns to user code execution (**go** command), the emulator uses the IRET instruction, which clears bit 15. The interrupt (NMI) and interrupt return (IRET) are transparent to the user except for the setting and clearing of bit 15.

The status of bit 15 is displayed with the **ds -p** command. The binary value of register symbol ICRIS is 00XX XXXX 0000 0XXX when bit 15 is cleared, and 10XX XXXX 0000 0XXX when bit 15 is set high.

PROM Programmer Considerations

Because of power supply limitations, the 80186/80188 Emulator must **not** be selected while you are programming a PROM.

Emulating the Execution Bus

Internally, the 80186 and 80188 are dual microprocessors. Each contains an Execution Unit (EU) and a Bus Interface Unit (BIU). The Bus Interface Unit prefetches instructions for the Execution Unit, and interfaces the EU to the outside world. The BIU communicates with the EU via an "Execution Bus".

The 80186/80188 Emulator emulates the information on the Execution Bus so that instructions are disassembled and traced accurately. Breakpoints are generated by information on the Execution Bus.

Prototype Interrupts During Trace All Mode

If a prototype interrupt occurs while operating in trace all mode, the instruction immediately preceding the interrupt may be displayed twice. The first display is the execution of the instruction, and the second display is caused by the prototype interrupt. The mnemonics of the second display are the same as for the first display. However, the register contents of the second display show the register changes that resulted from the interrupt. Even though this instruction is displayed twice, it is only executed once. If an optional TTA is available, the TTA **disp** command shows the correct sequence of events.

DMA Activity During Trace All Mode

When using the trace all mode during normal DMA transfers, in most cases the internal DMA registers are displayed for every other DMA transfer (one transferred byte is shown for every two transfers).

Short Instructions During Trace All Mode

When using the trace all mode, you might not see the trace displayed on even addresses for short instructions of two clock cycles. These instructions are executed even though they are not displayed.

TTA Display Changes During DMA Activity

The 80186/80188 is a pipe lined microprocessor and during DMA activity continues to execute the instructions that are in the queue, providing the execution does not intervene with the DMA activity. An example of a non-intervening instruction is one that requires a move from one register to another. However, if the first instruction fetched requires a move from a register to memory, the execution of the instruction is suspended until the DMA activity is completed.

During DMA activity, the TTA display may be affected as follows:

1. If the executing instruction causes no intervention of DMA activities, the first and second instruction fetches may be displayed separately in the TTA's display. The read/write DMA activities are shown between the two instruction fetches. This displayed separation of the instruction fetches is necessary because of the amount of time required to store the DMA activities.

NOTE

The displayed separation depends on several variables: the queue depth, the type of instruction, and the length of the DMA transfer.

2. If the executing instruction causes an intervention of DMA activities, the first instruction fetch is displayed. The second instruction fetch is not displayed until it is executed at the completion of the DMA activity. This prevents the TTA from correctly disassembling the instructions because of the time difference between the two fetches.

TTA Constraints

The 80186/80188 Emulator imposes the following constraints when using the optional TTA:

1. Supports only the CYC mode of operation for the **cons** command.
2. Supports an 8-bit value for the **data** command.

Locating the Stack Pointer

You must be careful where you locate the emulating processor's stack pointer (SP) in relationship to the PCB's base address. When a break is generated, the emulating processor pushes its registers onto the stack, creating six additional stack writes from the SP's address. (These stack pushes are not seen by the prototype circuitry.) If the SP's address is within six locations of the address area reserved by the PCB (256 bytes), the area is altered and could cause the emulating processor to crash.

For example, when the PCB's base address is set for 0FF00 in memory space, an address range of 0FF00-0FFFF is reserved for the PCB. If the SP is set to an address within the address range of 00000-00005, the next system break causes the emulating processor to push its registers into the area reserved for the PCB. In this example, the SP must be set for an address of 00006 or higher to prevent altering the PCB's area.

Segment Registers

The 80186/80188 microprocessor contains four segment registers: Code Segment (CS), Data Segment (DS), Stack Segment (SS), and Extra Segment (ES). Each segment register contains the 16 most significant bits of an 80186/80188 20-bit address. Addresses are given as a 16-bit offset to the segment base address.

NOTE

*Not all instructions that change the segment registers are displayed when you execute a program during trace all mode. The **tra all** command skips over these instructions.*

Register Symbols—CSX, DSX, SSX and ESX

A special symbol is assigned to each of the four memory segment registers that is 16 times the value in that segment's associated segment register. For example, the symbol DSX represents a value 16 times the value of the DS Register.

The following example shows a use for the CSX symbol.

Example

The 80186/80188 microprocessor creates the effective address (EA) for a symbolic label by adding 16 times the value of the current CS register to the contents of the IP register.

The 80186/80188 Emulator only knows the effective address of a symbol. However, the **address** parameter of the **g** command contains only IP information. In some cases, when the EA of a symbol is greater than FFFF or the CS is not zero, the 80186/80188 Emulator calculates an IP larger than 16 bits, and a truncation error occurs.

You can use the EA and the CS to calculate the IP.

```
IP = EA - (16 x CS)
```

Because the value of CSX is 16 times CS, you can avoid truncation errors by subtracting the CSX symbol in the expression that represents the desired address.

```
> g START-CSX
```

The **-CSX** modifier subtracts the CS value from the EA, ensuring that the IP contains no more than 16 bits.

SERVICE CALLS

Service calls (SVCs) enable your program to use many 8450, 8550, or 8560 system capabilities.

An SVC is invoked with an 80186/80188 OUT instruction followed by two NOPs. The OUT instruction's operand is indirectly referenced through Register DX when the SVC port range is larger than 00FF. (For example, the default SVC port range of 01000-01007 requires that the DX Register be used.) This instruction sequence directs the system to a specified memory address called the SRB (service request block) pointer. The SRB pointer points to the address of the SRB. The SRB pointer tells the system where to find the data stored in the SRB. The SRB tells the development system what service to perform. Refer to the service calls section of your System Users Manual for an explanation of service calls, service request blocks, and SRB pointers.

Table 2-19 shows the default addresses for the eight SRB pointers. These addresses and their associated port values can be altered with the **svc** command to suit your program requirements. See the Command Dictionary of your System Users Manual for syntax and use of the **svc** command.

SVCs in Modes 1 and 2

The 80186/80188 Emulator supports SVCs for all three emulation modes. In all three modes, use **two** NOPs following the I/O instruction. This allows enough time for the SVC to occur.

NOTE

In mode 1, only the instruction sequence that calls the SVC may reside in prototype memory. The SRB pointers, the SRB, and the optional I/O buffer must reside in program memory.

SRB Format

The 80186/80188 Emulator uses the LAS (Large Address Space) format for SRBs and SRB pointers. This format is described in the service calls section of your System Users Manual.

SVC Demonstration

Figure 2-27 lists an 80186/80188 program that uses four SVC functions: Assign Channel, Read ASCII, Write ASCII, and Abort. The program's algorithm is explained in the service calls section of your System Users Manual, which demonstrates a version of the program written in 8085A assembly language. Using the program in Figure 2-27, you can perform a parallel demonstration with the 80186/80188 B Series Assembler and 80186/80188 Emulator.

The program accepts a line of ASCII characters from the system terminal. When it receives a RETURN character, the program writes the line to the line printer and accepts another line. On the 8550, output to the line printer is buffered. No text is printed until the 8550's line printer buffer becomes full or the program ends.

Table 2-19
80186/80188 Service Calls

SVC Number	Service Call ^a		Default Location of SRB Pointer
	Mnemonic	Hexadecimal	
1	MOVW DX,#01007H	BA0710	40-43
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
2	MOVW DX,#01006H	BA0610	44-47
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
3	MOVW DX,#01005H	BA0510	48-4B
	OUT DX,AL	EE	
	NO	90	
	NOP	90	
4	MOVW DX,#01004H	BA0410	4C-4F
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
5	MOVW DX,#01003H	BA0310	50-53
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
6	MOVW DX,#01002H	BA0210	54-57
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
7	MOVW DX,#01001H	BA0110	58-5B
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	
8	MOVW DX,#01000H	BA0010	5C-5F
	OUT DX,AL	EE	
	NOP	90	
	NOP	90	

^a The default SVC port range, 01000-01007, is assumed.

NOTE

The program shown in Figure 2-27 is written for a B Series assembler. To make this program acceptable for the A Series assembler used on some 8550s, change each single quote (') to a double quote ("). To terminate the program, enter a CTRL-Z while the program is waiting for input.

```

; SSSSS V V CCCCC
; S V V C
; SSSSS V V C DEMONSTRATION. 80186/80188 EMULATOR
; S V V C
; SSSSS V CCCCC
      ORG 40H ; Beginning of SRB vector
      BYTE 0,BITS(SRB1FN,16,4),HI(SRB1FN),LO(SRB1FN)
      BYTE 0,BITS(SRB2FN,16,4),HI(SRB2FN),LO(SRB2FN)
      BYTE 0,BITS(SRB3FN,16,4),HI(SRB3FN),LO(SRB3FN)
      BYTE 0,BITS(SRB4FN,16,4),HI(SRB4FN),LO(SRB4FN)
      BYTE 0,BITS(SRB5FN,16,4),HI(SRB5FN),LO(SRB5FN)
;      End of SRB vector
      ORG 60H ; Set up SRB areas
;      SRB1 = Assign 'CONI' to Channel 0
SRB1FN  BYTE 10H ; Assign
        BYTE 00H ; to Channel 0
SRB1ST  BLOCK 01H ; Status returned here
        BLOCK 03H ; Bytes 4 through 6 not used
        BYTE 00H,05H ; Length of 'CONI' +
        BYTE 00H ; Pointer
        BYTE BITS(CONI,16,4) ; to
        BYTE HI(CONI) ; 'CONI'
        BYTE LO(CONI) ; +
;      End of SRB1
;      SRB2 = Assign 'LPT' to Channel 1
SRB2FN  BYTE 10H ; Assign
        BYTE 01H ; to Channel 1
SRB2ST  BLOCK 01H ; Status returned here
        BLOCK 03H ; Bytes 4 through 6 not used
        BYTE 00H,04H ; Length of 'LPT' +
        BYTE 00H ; Pointer
        BYTE BITS(LPT,16,4) ; to
        BYTE HI(LPT) ; 'LPT'
        BYTE LO(LPT) ; +
;      End of SRB2
;      SRB3 = Read ASCII line from CONI (Channel 0)
SRB3FN  BYTE 01H ; Read ASCII
        BYTE 00H ; from Channel 0
SRB3ST  BLOCK 01H ; Status returned here
        BLOCK 01H ; Byte 4 not used
        BLOCK 02H ; Byte count returned here
        BYTE 01H,00H ; 256 Bytes in our buffer
        BYTE 00H ; Pointer
        BYTE BITS(BUFFER,16,4); to
        BYTE HI(BUFFER) ; BUFFER
        BYTE LO(BUFFER) ; +
;      End of SRB3

```

Figure 2-27. 80186/80188 SVC demonstration program listing (Part 1).

```

;      SRB4 = Write ASCII line to LPT (Channel 1)
SRB4FN BYTE 02H          ; Write ASCII
      BYTE 01H          ; to Channel 1
SRB4ST BLOCK 01H         ; Status returned here
      BLOCK 01H         ; Byte 4 not used
      BLOCK 02H         ; Byte count returned here
      BYTE 01H,00H      ; 256 bytes in our buffer
      BYTE 00H          ; Pointer
      BYTE BITS(BUFFER,16,4); to
      BYTE HI(BUFFER)   ; BUFFER
      BYTE LO(BUFFER)   ; +
;      End of SRB4
;      SRB5 = Abort (Close all channels and terminate)
SRB5FN BYTE 1FH          ; Abort
      BLOCK 0BH         ; Bytes 2 through 12 not used
;      End of SRB5
;
BUFFER BLOCK 100H        ; Our I/O area
CONI   ASCII 'CONI'      ; ASCII of 'CONI'
      BYTE 0DH           ; +
LPT    ASCII 'LPT'       ; ASCII of 'LPT'
      BYTE 0DH           ; +
;      End of data definitions
;
;      Beginning of executable code
START  ORG 1000H          ; Entry point into program
      MOVW DX,#01007H    ; Call SVC1
      OUT DX,AL          ; to assign 'CONI'
      NOP                ; to Channel 0
      NOP                ;
      MOV AL,SRB1ST      ; Check the status
      CMP AL,#00H        ; to see if all went well
      JNZ ABORT          ; No? Stop everything
      MOVW DX,#01006H    ; Call SVC2
      OUT DX,AL          ; to assign 'LPT'
      NOP                ; to Channel 1
      NOP                ;
      MOV AL,SRB2ST      ; Check the status
      CMP AL,#00H        ; to see if all went well
      JNZ ABORT          ; No? Stop everything

```

Figure 2-27. 80186/80188 SVC demonstration program listing (Part 2).


```
AL00P  MOVW    DX,#01005H      ; Call SVC3
      OUT      DX,AL          ; to read a line
      NOP                      ; from 'CONI'
      NOP                      ; into the buffer
      MOV      AL,SRB3ST      ; Check the status
      CMP      AL,#00H        ; to see if all went well
      JNZ      ABORT          ; No? Stop everything
      MOVW     DX,#01004H      ; Call SVC4
      OUT      DX,AL          ; to write the line
      NOP                      ; from the buffer
      NOP                      ; to 'LPT'
      MOV      AL,SRB4ST      ; Check the status
      CMP      AL,00H         ; to see if all went well
      JZ       AL00P          ; Yes? Back to read another line
      ; No? Fall through to termination
ABORT  MOVW     DX,#01003H      ; Call SVC5
      OUT      DX,AL          ; to exit
      NOP                      ; the program
      NOP                      ;
      END      START          ; End of program
```

Figure 2-27. 80186/80188 SVC demonstration program listing (Part 3).

Section 3

EMULATOR DEMONSTRATION RUN

INTRODUCTION

This section contains a demonstration run that shows you how to load, execute, and monitor a simple 80186 program on your 8540 or 8550. This demonstration run is designed to be used with the Learning Guide of your System Users Manual. Before you perform this demonstration, your 80186/80188 Emulator hardware and control software must be installed in your development system. Refer to Section 6 of this manual for hardware and software installation procedures.

NOTE

The 8550 development system does not have an 80186/80188 Assembler. If you have an 8086/8088 Assembler for the 8550, you can use the demonstration program and control software provided with your 8086/8088 Emulator to assemble and load the demonstration program used in this section. The demonstration program in this section does not contain the unique instructions for the 80186/80188 microprocessor.

DEMONSTRATION PROGRAM

Figure 3-1 shows the source code and object code for the demonstration program. Refer to this figure as you examine the demonstration program.

Examine the Demonstration Program

The demonstration program adds five numbers from a table stored in locations 500-504 in program memory and leaves the sum in Register AL. You will place values in the table later in this demonstration. The 8085A Emulator demonstration run in your System Users Manual contains a flowchart that illustrates the steps of this program.

The source code contains two kinds of statements: assembler directives (like ORG and BYTE) and 80186 assembly language instructions. The assembler directives are microprocessor-independent and are explained in the 8085A Emulator demonstration run. The 80186 assembly language instructions are discussed in the following paragraphs.

Set the Table Pointer

The MOVW instruction shown below, loads the address of the table (500) into Register BX. Register BX then points to the first element of the table. The END directive uses the label START to specify that the MOVW instruction is the first instruction executed.

```
MOVW BX,#TABLE
```

```

01          ;80186 DEMONSTRATION RUN PROGRAM
02          SECTION DEMO
03          ORG    100H          ;START PROGRAM CODE AT ADDRESS 100
04 00000100 BB0005  START  MOVW  BX,#TABLE  ;SET TABLE POINTER
05 00000103 B90500          MOVW  CX,#TSIZE  ;SET PASS COUNTER
06 00000106 32C0          XORB  AL,AL      ;CLEAR ACCUMULATOR
07 00000108 0207  ALOOP  ADDB  AL,[BX]     ;ADD BYTE FROM TABLE AND
08 0000010A 43          INCW  BX          ; POINT TO NEXT BYTE
09 0000010B E2FB          LOOP  ALOOP      ;DECREMENT CX AND LOOP UNTIL CX=0
10 0000010D BAO710          MOVW  DX,#01007H ;I/O ADDRESS FOR EXIT SVC
11 00000110 EE          OUT   DX,AL       ;CALL EXIT SVC
12 00000111 90          NOP              ; TO END PROGRAM
13 00000112 90          NOP              ; EXECUTION
14          ;SRB POINTER
15          ORG    40H          ;STORE SRB POINTER AT ADDRESS 40
16 00000040 00000044      BYTE  OH,OH,OH,44H ;POINT TO SRB FOR EXIT SVC
17          ;SRB FOR EXIT SVC
18 00000044 1A          BYTE  1AH          ;1AH = FUNCTION CODE FOR EXIT SVC
19          ;TABLE OF NUMBERS TO BE ADDED
20          TSIZE  EQU    5          ;TABLE SIZE = 5
21          ORG    500H          ;SET UP TABLE AT ADDRESS 500
22          TABLE BLOCK  TSIZE
23          LIST   DBG
24          END    START

```

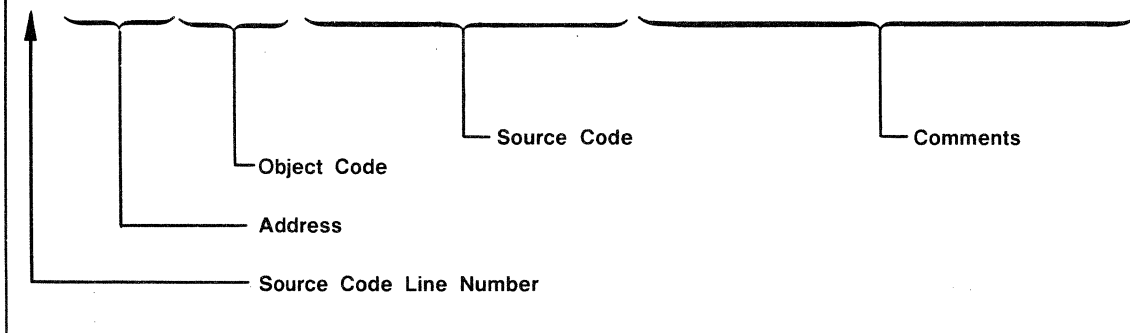


Figure 3-1. 80186 demonstration run program.

Set the Pass Counter

The MOVW instruction shown below, moves the value of TSIZE into CX. Register CX is then used as the pass counter. This instruction sets the number of passes to 5.

```
MOVW  CX,#TSIZE
```

Clear the Accumulator

The XORB instruction shown below, zeros Register AL. This allows you to use AL as the accumulator and start adding numbers from the table into AL.

```
XORB  AL,AL
```

Add a Byte from the Table

The ADDB instruction shown below, adds the byte addressed by the BX register into the accumulator AL. The label ALOOP represents this instruction's address. The LOOP instruction uses this label.

```
ADDB  AL,[BX]
```

Point to the Next Byte

The INCW instruction shown below, increments register BX. Register BX then points to the next byte in the table. For example, register BX was initialized to contain 500. The first time this instruction is executed, register BX contains 501, the address of the second element of the table.

```
INCW  BX
```

Decrement CX and Loop Until CX = 0

The LOOP instruction shown below, decrements Register CX, the pass counter. If CX is not yet 0, the program jumps back to the label ALOOP. If CX is equal to 0, the program calls the Exit SVC routine.

```
LOOP  ALOOP
```

Call Exit SVC

The last four instructions of the program are shown below. These instructions constitute a service call (SVC) that causes an exit from the program. For more information on SVCs, refer to "Service Calls" in Section 2 of this manual.

```
MOVW  DX,#01007H
OUT   DX,AL
NOP
NOP
```

DEVELOPMENT SYSTEM CONFIGURATIONS

Figure 3-2 shows the various hardware configurations that apply to this demonstration run. These configurations are briefly described here with detailed descriptions later in this section. Find the case that is appropriate for your hardware configuration.

1. **Case 1: For 8550 users.** This demonstration shows you how to assemble the 80186/80188 program on your 8550. If your system disk does not contain an 8086/8088 Assembler, you cannot complete this part of the demonstration. Skip ahead in this section to the detailed description "Case 4: Patch the Program into Memory" for instructions.
2. **Case 2: For 8540/8560 or 8550/8560 users.** If you have an 8540/8560 or 8550/8560 system and your 8560 has an 80186/80188 Assembler installed, you can create and assemble the program on the 8560 and then download it to the 8540 or 8550. This demonstration shows how to download from this configuration to the 8540 or 8550.
3. **Case 3: For 8540 or 8550 users with host computers other than the 8560.** If you have an 8540 or 8550 that is connected to a host computer other than an 8560, this manual doesn't give you a specific list of commands for creating and assembling the program on your host. The demonstration shows you how to create the Tekhex file using your host's assembler or text editor, then download the file to the 8540 or 8550 through the system's COM interface.
4. **Case 4: For other hardware configurations.** If none of these cases applies to you, this demonstration shows how to patch the program into memory using the **p** command.

ASSEMBLE AND LOAD THE DEMONSTRATION PROGRAM

Now you can create the program to run on your emulator. Detailed discussions for each hardware configuration are contained in the following paragraphs. Work through the discussion appropriate for you. Once the program is loaded or patched into memory, you can then execute the program on your emulator. Turn to the heading "Run the Demonstration Program," later in this section for instructions to execute the program.

Case 1: Assemble and Load on the 8550

NOTE

The 8550 development system does not have an 80186/80188 Assembler. If you have an 8086/8088 Assembler for the 8550, you can use the demonstration program and control software provided with your 8086/8088 Emulator to assemble and load the demonstration program used in this section. The demonstration program in this section does not contain the unique instructions for the 80186/80188 microprocessor.

This discussion shows you how to copy (from your 80186/80188 Emulator software installation disk), assemble, and load the demonstration program into 8550 program memory. If your system disk does not contain an 8086/8088 Assembler, you cannot copy the program or complete this part of the demonstration. Skip ahead in this section to the description "Case 4: Patch the Program into Memory" for instructions.

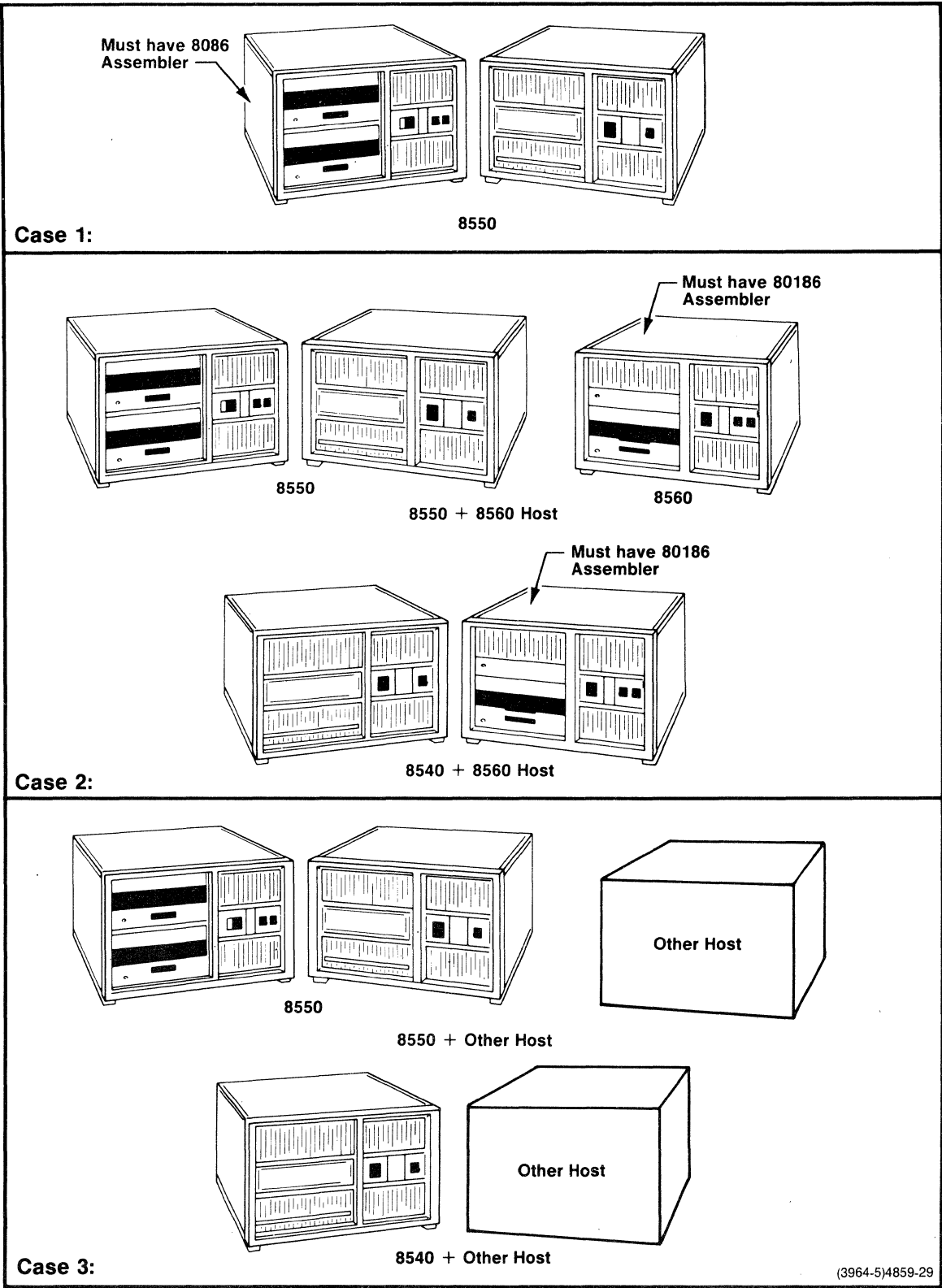


Figure 3-2. Development system configurations.

Start Up and Log In

Turn on your 8550 system. For start up instructions, refer to the paragraph "Start Up the 8550 and Its Peripherals" in the Learning Guide of your System Users Manual. Place your system disk in drive 0 and shut the drive 0 door. When your system displays the `>` prompt, place your 80186/80188 Emulator software installation disk in drive 1 and shut the drive 1 door.

Use the **dat** command to set the current date and time. For example, for 2:30 p.m. on March 30, 1984, enter the following command line:

```
> dat 30-mar-84/2:30 pm
```

Use the **sel** command to tell DOS/50 to use the assembler and emulator software designed for the 8086/8088 microprocessors.

```
> sel 8086
```

The system responds with the current version number:

```
8086 Emulator V n.nn mm/dd/yy
```

The **sel** command automatically sets the emulation mode to 0.

Copy the Demonstration Run Program from the Installation Disk

Enter the following command lines to create an empty directory called *DEMO* on your system disk and make *DEMO* the current directory. The **br** command creates a brief name, *root*, to mark the old current directory. At the end of this demonstration, you will return to this *root* directory and delete the *DEMO* directory and its contents.

```
> br root /usr
```

```
> create DEMO
```

```
> user DEMO
```

Now use the **cop** command to copy all the files in the *DEMO2* directory on the installation disk into the *DEMO* directory you just created:

```
> cop /vol/emu.8086/DEM02/ * *
```

Remove your installation disk from drive 1 and put the disk away.

Now list the files you just copied to the current directory:

```
> l
```

```
FILENAME
```

```
ASM
```

```
LOAD
```

```
Files used      124
Free files      132
Free blocks     821
Bad blocks      0
```

The file named *ASM* contains the assembly language source code for this demonstration program, and the file named *LOAD* contains the executable object code. This copy of *LOAD* will be used in the demonstration only if you do not have an 8086/8088 Assembler and cannot create your own object file and load file from the source file.

Examine the Demonstration Program

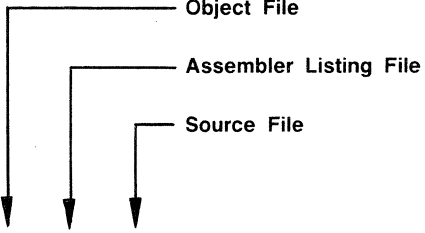
Enter the following command line to display the source file *ASM* on the system terminal. Your display may differ slightly from this example because of your terminal's tab settings.

```
> con ASM
;80186 DEMONSTRATION RUN PROGRAM
SECTION DEMO
    ORG    100H        ; START PROGRAM CODE AT ADDRESS 100 HEX
START  MOVW  BX,#TABLE  ; SET TABLE POINTER
      MOVW  CX,#TSIZE   ; SET PASS COUNTER
      XORB  AL,AL       ; CLEAR ACCUMULATOR
ALOOP  ADDB  AL,[BX]    ; ADD BYTE FROM TABLE
      INCW  BX         ; POINT TO NEXT BYTE
      LOOP  ALOOP      ; DECREMENTS CX AND LOOPS UNTIL CX=0
      MOVW  DX,#01007H ; I/O ADDRESS FOR EXIT SVC
      OUT   DX,AL      ; CALL EXIT SVC
      NOP                    ; TO END PROGRAM
      NOP                    ; EXECUTION
;SRB POINTER
    ORG    40H        ; STORE SRB POINTER AT ADDRESS 40 HEX
    BYTE   0H,0H,0H,44H ;POINT TO SRB FOR EXIT SVC
;SRB FOR EXIT SVC
    BYTE   1AH        ; 1AH = FUNCTION CODE FOR EXIT SVC
;TABLE OF NUMBERS TO BE ADDED
TSIZE  EQU    5        ; TABLE SIZE = 5
    ORG    500H       ; SET UP TABLE AT ADDRESS 500 HEX
TABLE  BLOCK  TSIZE
      LIST  DBG
      END   START
```


Assemble the Source Code

If you do not have an 8086/8088 Assembler on your system disk, you cannot perform this step. Skip the next four commands: **asm**, **cop**, **link**, and **l**.

The **asm** (assemble) command translates assembly language (source code) into binary machine language (object code). The **asm** command also creates an assembler listing that correlates the object code with the source code. Enter the following command line to assemble the source code in the file *ASM* and create the listing and object files *ASML* and *OBJ*:



```
> asm OBJ ASML ASM
```

Tektronix 8086 Vxx.xx-xx Copyright (c) 1981 Tektronix
**** Pass 2
24 Source Lines 24 Assembled Lines xxxxx Bytes Available
>>> No assembly errors detected <<<

Make sure the printer is properly connected and is turned on. Then enter the following command to copy the assembler listing to the line printer:

```
> cop ASML lpt
```

The different fields of your source listing are shown in Figure 3-1, earlier in this demonstration. For a detailed explanation of assembler listings, consult your Assembler Core Users Manual.

Link the Object Code

The linker creates an executable load file from one or more object files. Enter the following linker command to create a load file called *LOAD* from your object file, *OBJ*:

```
> link -O OBJ -o LOAD -d
```

The linker command options **-O** and **-o** specify the object file and load file respectively. The **-d** command option causes the linker to pass the program symbols from the object file to the load file for use in program debugging.

The files generated by the **asm** and **link** commands should now be on your disk. Enter the following command to list the files in your current directory:

```
> l
FILENAME

ASM
LOAD
OBJ
ASML

Files used      126
Free files      130
Free blocks     811
Bad blocks      0
```

Four files are now listed in your directory. The files *OBJ* and *ASML* were created by the assembler, and *LOAD* was created by the linker.

Select the 80186/80188 Emulator

Before loading the demonstration program into program memory, use the **sel** command to turn off the 8086 Emulator/Assembler and select the 80186 Emulator. Enter the **sel** command as follows:

```
> sel 80186
```

Load the Program into Memory

Now you can load the object code from the load file *LOAD* into program memory.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000-00FFF.

```
> al 0
```

Zero Out Memory. Before you load any code, use the **f** (fill) command to fill program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40-11F with zeros:

```
> f 40 11f 00
```

Check that Memory is Filled with Zeros. Check the contents of memory with the **d** (display) command. The display shows the data in hexadecimal format, and also shows the corresponding ASCII characters. Display the contents of memory addresses 40–11F with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Load the Object Code into Memory. Load the object code for the demonstration program (contained in the load file *LOAD*) into program memory as shown here:

```
> lo <LOAD
```

Load the Program Symbols. The source code for the demonstration program contains the directive **LIST DBG** (refer to Figure 3-1). Because of this directive, the object file contains a list of the symbols that appeared in the source code, and the values associated with those symbols. Because you included the **-d** command option when you invoked the linker, those symbols were passed to the load file. Use the **symlo** command to load those symbols into the symbol table in 8550 system memory.

```
> symlo -s <LOAD
```

The **-s** option means that both addresses and scalars are loaded. If you omit the **-s** option, only addresses are loaded. A scalar is a number that is not an address. In Figure 3-1, the scalar **TSIZE** represents the length of a table.

Later in this demonstration, whenever you use a symbol in an 8550 command line, DOS/50 refers to the symbol table you just created to find the value that the symbol represents.

You have now assembled and linked the demonstration program and loaded it into memory. You are now ready to run the program. Skip ahead in this section to “Run the Demonstration Program.”

Case 2: Assemble on the 8560; Download to the 8540 or 8550

This discussion shows you how to create the demonstration program source code and assemble it on an 8560, then download it into the 8540 or the 8550 program memory. If your 8560 does not have an 80186/80188 Assembler, you cannot complete this part of the demonstration. Skip ahead in this section to the description “Case 4: Patch the Program into Memory” for instructions.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550 connected to an 8560. You can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

Start Up and Log In

Start up your 8540, make sure it's in TERM mode, and log in to the 8560 TNIX operating system. See your 8560 System Users Manual for details.

While you are logged in to TNIX, your system prompt is \$. Later in this demonstration (Case 3 and Case 4) the system prompt > appears for the 8540s and 8550s in LOCAL mode. Every command you enter is processed by TNIX. If you enter an OS/40 command, TNIX passes the command to the 8540.

Enter the following commands to select the 80186/80188 Assembler on the 8560 and the 80186/80188 Emulator in the 8540:

```
$ uP=80186; export uP
$ sel 80186
```

NOTE

*If you are using an 8550, enter **sel 80186** or **sel 80188** depending on which prototype control probe you have installed.*

The **sel** command automatically sets the emulation mode to 0.

Create the Demonstration Program

Enter the following TNIX command lines to create an empty working directory called *demo*. You'll create your source file and related files in this *demo* directory.

```
$ mkdir demo
$ cd demo
```

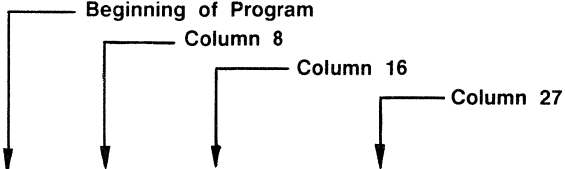
Now use the TNIX editor, **ed**, to create the demonstration program source file. The following command line invokes the editor and specifies that you want to create a file called *asm*:

```
$ ed asm
?asm
```

The editor responds with “?asm” to remind you that the file *asm* does not exist. The editor does **not** give a prompt to let you know that the editor is ready for input.

Enter the Text. Now enter the editor command **a** (append text) and type in the program. Use the BACKSPACE key to erase typing mistakes.

a <CR>



```

;80186 DEMONSTRATION RUN PROGRAM
SECTION DEMO
ORG     100H      ;START PROGRAM CODE AT ADDRESS 100
START  MOVW      BX,#TABLE ;SET TABLE POINTER
MOVW   CX,#TSIZE  ;SET PASS COUNTER
XORB   AL,AL      ;CLEAR ACCUMULATOR
ALOOP  ADDB      AL,(BX)   ;ADD BYTE FROM TABLE AND
      INCW      BX        ; POINT TO NEXT BYTE
      LOOP     ALOOP      ;DECREMENTS CX AND LOOPS UNTIL CX=0
      MOVW     DX,#01007H ;I/O ADDRESS FOR EXIT SVC
      OUT      DX,AL      ;CALL EXIT SVC
      NOP      ; TO END PROGRAM
      NOP      ; EXECUTION

;SRB POINTER
ORG     40H      ;STORE SRB POINTER AT ADDRESS 40
      BYTE     OH,OH,OH,44H ;POINT TO SRB FOR EXIT SVC

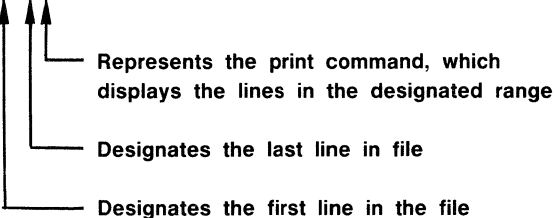
;SRB FOR EXIT SVC
      BYTE     1AH      ;1AH = FUNCTION CODE FOR EXIT SVC

;TABLE OF NUMBERS TO BE ADDED
TSIZE  EQU       5      ;TABLE SIZE = 5
ORG     500H      ;SET UP TABLE AT ADDRESS 500
TABLE  BLOCK     TSIZE
      LIST      DBG
      END       START
. <CR>
  
```

At the end of your text, enter a period on a line by itself. The editor will return to the command mode.

Check for Errors. The following editor command displays the text you have entered. Check for typing mistakes.

l,\$p <CR>



l,\$p <CR>

- Represents the print command, which displays the lines in the designated range
- Designates the last line in file
- Designates the first line in the file

Correct any mistakes with the text editor **ed**. If you're not familiar with **ed**, Table 3-1 lists the commands you need to add, delete, or replace lines. For more information about **ed**, refer to your 8560 System Users Manual.

Table 3-1
Basic 8560 Editing Commands

Command	Function
mm,nnp <CR>	Displays lines mm through nn.
nn <CR>	Makes line nn the current line.
d <CR>	Deletes the current line.
a <CR> text . <CR>	Adds text after the current line. (Enter a period on line by itself.)
c <CR> text . <CR>	Replaces the current line with the text you type in. (Enter a period on line by itself.)

When your text is correct, enter the **w** command to write the text to the source file *asm*:

```
w <CR>
902
```

The editor responds with the number of characters it wrote to the file.

Finally, enter the **q** command to quit the editor and return to TNIX:

```
q <CR>
$ ← TNIX Prompt
```

Assemble the Source Code

The TNIX **asm** (assemble) command translates assembly language (source code) into binary machine language (object code). The **asm** command also creates an assembler listing which can be used to correlate the object code with the source code. Enter the following command line to assemble the source code in the file *asm* and create the listing and object files *asml* and *obj*:

```
$ asm obj asml asm
```

```
Tektronix ASM 80186
Vxx.xx-xx (8560 )
*****Pass 2

24 Lines Read
24 Lines Processed
0 Errors
```

Enter the following command to print the assembler listing on the 8560's line printer:

```
$ lp1r asml
```

Check page 1 of your listing. You should have no error message from the assembler. If your source code contains errors, follow these steps:

1. Refer to your Assembler Users Manual to find out what the error messages mean.
2. Enter the command **ed asm** to get back into the editor and fix the mistakes in your source code. Exit the editor with the **w** and **q** commands.
3. Enter the command **asm obj asml asm** to reassemble your source code.

Link the Object Code

The linker creates an executable load file from one or more object files. Enter the following command to create a load file called *load* from your object file *obj*. Be sure to capitalize parameters as shown.

```
$ link -d -O obj -o load
```

The **-d** option causes the linker to pass the program symbols from the object file to the load file for program debugging.

The files generated by the **asm** and **link** commands should now be in your working directory *demo*. Enter the following command to list the files in your working directory:

```
$ ls
asm
asml
load
obj
```

Four files are now listed in your directory. The files *obj* and *asml* were created by the assembler, and the file *load* was created by the linker.

Download the Program to the 8540

You can now download the object code produced by the 8560's linker into 8540 program memory.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000-00FFF.

```
$ al 0
```

Zero Out Memory. Before you download any code, use the OS/40 **f** (fill) command to fill 8540 program memory with zeros. Later, when you examine the memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40-11F with zeros:

```
$ f 40 11f 00
```

Check that Memory is Filled with Zeros. Check the contents of memory with the OS/40 **d** (display) command. The display shows the data in hexadecimal format and also shows the corresponding ASCII characters. Display the contents of memory addresses 40-11F with the following command line:

```
$ d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```


Download the Object Code. Enter the following command line to download the object code from the 8560 load file *load* to the 8540 program memory:

```
$ lo <load
```

Download the Program Symbols. The source code for the demonstration program contains the directive LIST DBG (shown in Figure 3-1). Because of this directive, the object file contains a list of the symbols that appear in the source code, and the values associated with those symbols. Because you included the **-d** option in the link command line, those symbols were passed to the load file. Use the OS/40 **symlo** command to download those symbols into the symbol table in 8540 system memory.

```
$ symlo -s <load
```

The **-s** option means that both addresses and scalars are downloaded. If you omit the **-s** option, only addresses are downloaded. A scalar is a number that is not an address. In Figure 3-1, the scalar TSIZE represents the length of a table.

Later in this demonstration, whenever you use a symbol in an 8540 command line, OS/40 refers to the symbol table you just created to find the value that the symbol represents.

You have now assembled and linked the demonstration program and downloaded it into memory. You are now ready to run the program. Skip ahead in this section to “Run the Demonstration Program.”

Case 3: Download from Your Host to the 8540 or 8550

This discussion gives some general instructions for downloading the demonstration program from an unspecified host computer to the 8540 or the 8550 program memory. If your 8540 is not equipped with the optional COM interface package, you cannot complete this part of the demonstration. Skip ahead in this section to the description "Case 4: Patch the Program into Memory" for instructions. The COM interface software is standard on the 8550.

Since your host computer is not an 8560, this manual only provides a general outline for creating the demonstration program and downloading it to the 8540 or the 8550. Once you have determined the command sequence appropriate for your host, record this information in the space provided in Figure 3-3.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550 connected to an unspecified host computer. You can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

<u>Create the Extended Tekhex Load Module</u> <u>Start Up and Log In</u> (Start up the 8540.) <pre>> sel 80186 > al 0 > f 40 11f 00 > d 40 11f</pre> <u>Establish Communication</u> <u>Download the Load Module</u> <u>Terminate Communication</u>
--

Figure 3-3. Host computer commands for preparing the demonstration program.

Create the Extended Tekhex Load Module

To download the object code to the 8540, the code must be in extended Tekhex format, as shown in Figure 3-4. You can create the load module in one of two ways:

1. Use your host computer's text editor and type in the load module.
2. Use your host computer's 80186/80188 Assembler:
 - a. Translate the demonstration program into the language of your host's 80186/80188 Assembler.
 - b. Create and assemble the source file.
 - c. Link the object code, if necessary.
 - d. Translate the object code produced by the assembler or linker into extended Tekhex format. The intersystem communication section of your System Users Manual provides a general algorithm for conversion to extended Tekhex format.

Figure 3-4A shows an extended Tekhex load module that contains the object code and program symbols for the demonstration program. Figure 3-4B gives the meanings of the different fields in the message blocks. If you have a host computer other than an 8560, you can create this load module and download it to your 8540 or 8550.

Start Up and Log In

Start up your 8540 and enter the following command to select the 80186 Emulator:

```
> sel 80186
```

NOTE

*If you are using an 8550, enter either **sel 80186** or **sel 80188** depending on which prototype control probe you have installed.*

The **sel** command automatically sets the emulation mode to 0 and displays the current version number.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000-00FFF.

```
> al 0
```

Zero Out Memory. Before you download any code, use the OS/40 **f** (fill) command to fill 8540 program memory with zeros. Later, when you examine the memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40-11F with zeros:

```
> f 40 11f 00
```

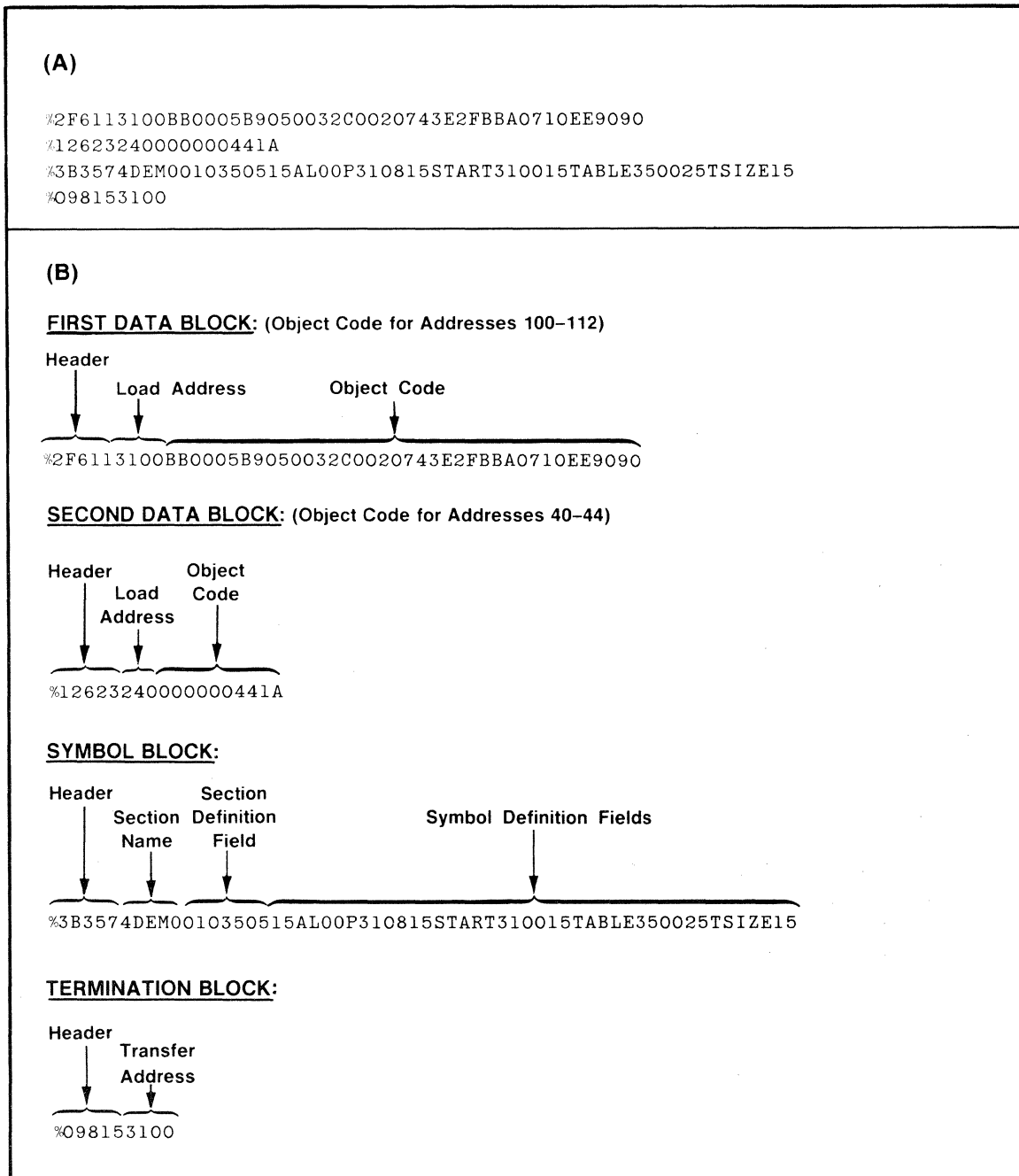


Figure 3-4. 80186 demonstration run program: extended Tekhex format.

Check that Memory is Filled with Zeros. Check the contents of memory with the OS/40 **d** (display) command. The display shows the data in hexadecimal format and also shows the corresponding ASCII characters. Display the contents of memory addresses 40–11F with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Download the Load Module. Be sure that your 8540 and your host computer are connected through an RS-232-C compatible communications link. Then perform the following steps to download the Tekhex load module to the 8540 program memory. Refer to the intersystem communication section of your System Users Manual to determine the commands and parameters appropriate for your host computer.

1. Enter the 8540 **com** command to establish communication. The parameters of the **com** command are host-specific. Log on to your host and execute any necessary host initialization commands.
2. Enter the command line that downloads the Tekhex load module to the 8540. This command line consists of a host computer command that performs the download, followed by a null character (CTRL-@ on most terminals) and a carriage return. The **com** command places the object code in 8540 program memory and puts the program symbols into the symbol table in 8540 system memory.
3. Log off from your host and terminate the **com** command execution by entering the null character and then pressing the ESC key.

Once you have downloaded the program to the 8540, you are now ready to run the program. Skip ahead in this section to "Run the Demonstration Program."

Case 4: Patch the Program into Memory

This discussion shows you how to patch the demonstration program into 8540 or 8550 program memory using the **p** command, and how to add the program symbols into the symbol table using the **adds** command.

Ordinarily, you would load the object code and symbols from a binary or hexadecimal load file, as illustrated for cases 1, 2, and 3. The procedure presented here is **not** normally used for preparing a program for execution. Use this procedure only if you have no standard means for preparing the program but would still like to try out the demonstration program.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550. You can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

Start Up and Log In

Start up your 8540 and enter the following command to select the 80186/80188 Emulator:

```
> sel 80186
```

NOTE

*For 8550 users, enter **sel 80186** or **sel 80188**, depending on which prototype control probe you have installed.*

The **sel** command automatically sets the emulation mode to 0 and displays the current version number.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000-00FFF.

```
> al 0
```

Zero Out Memory. Before you patch in any code, use the OS/40 **f** (fill) command to fill 8540 program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Enter the following command line to fill memory addresses 40-11F with zeros:

```
> f 40 11f 00
```

> d 40 11f

[illegible]

Diagram illustrating the assembly instructions and their corresponding patch addresses:

- 100: MOVW BX,#TABLE (First program instruction)
- BB0005: MOVW CX,#TSIZE (Second program instruction)
- B90500: XORB AL,AL (Third program instruction)
- 32C0: MOVW CX,#TSIZE (Fourth program instruction)

Patch Address

> p 108 020743E2FB

> p 10D BA0710EE9090

```
> p 40 000000441A
```

3-22

Put Symbols into the Symbol Table. Later in this demonstration, you will use symbols from the demonstration program (START, LOOP, TSIZE, and TABLE) when communicating with OS/40. When you use a symbol in a command line, OS/40 consults a symbol table in 8540 system memory to find the values that the symbol represent. Enter the following command line to add the program symbols and their values to the symbol table:

```
> adds START=100 AL00P=108 -s TSIZE=5 TABLE=500
```

The **adds** command cannot provide all the symbol-related information provided by the **symlo** command in cases 1 and 2 or by the **com** command in Case 3. Because this information is missing, some of the displays you produce later in this demonstration will not match the symbolic displays shown in this manual. For more information on the **adds** command, refer to the Command Dictionary of your System Users Manual.

You have patched the demonstration program into program memory and placed the program symbols in the symbol table. You can now run the program.

RUN THE DEMONSTRATION PROGRAM

For the rest of this demonstration, the commands you enter are shown in lowercase. If you are using an 8540 or an 8550, you may enter commands in lowercase or uppercase. If you **are** using an 8560, you **must** enter the name of every command in lowercase. In addition, the 8560's system prompt is \$.

Now that you have loaded the program into memory, you need to:

- Verify that the program was loaded correctly
- Put values in the table in memory for the program to add

Check Memory Contents Again

Before you loaded the program, you filled memory locations 40–11F with zeros. Look at the same memory area again with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 44 1A 00 00 00 00 00 00 00 00 00 00 00 ...D.....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 BB 00 05 B9 05 00 32 C0 02 07 43 E2 FB BA 07 10 .....2...C.....
000110 EE 90 90 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

The object code is loaded in two different blocks:

- The 80186 machine instructions are loaded at address 100 (specified by the first ORG directive in the source code).
- The information for the Exit SVC is loaded at address 40 (specified by the second ORG directive).

The contents of the table at address 500 are still undefined, but you will define these values soon under the heading “Put Values in the Table in Memory”.

Turn On the Symbolic Display

Enter the following command to tell the system to replace the hexadecimal display numbers with symbols from your program where it is appropriate:

```
> symd on
```

Disassemble the Object Code

The **di** (disassemble) command displays memory contents in hexadecimal notation and in assembly language mnemonics. You can use the **di** command to verify that the object code in memory corresponds to your source code. Enter the following command to disassemble the memory area occupied by the executable part of your program:

```
> di 100 112
LOC      INST      MNEM      OPER

START
000100   BB0005      MOVW      BX,#0500

DEMO+000103
000103   B90500      MOVW      CX,#0005

DEMO+000106
000106   32C0        XORB      AL,AL

ALOOP
000108   0207        ADDB      AL,[BX]

DEMO+00010A
00010A   43           INC       BX

DEMO+00010B
00010B   E2FB        LOOP      $-03

DEMO+00010D
00010D   BA0710      MOVW      DX,#1007

DEMO+000110
000110   EE           OUT       DX,AL

LOC      INST      MNEM      OPER

DEMO+000111
000111   90           NOP

DEMO+000112
000112   90           NOP
```

Compare the **di** command display with the assembler listing you generated earlier, or refer to Figure 3-1 for the source code in the demonstration program.

The memory area your program uses (location 0 through 504 at the end of the table) belongs to section DEMO. DEMO was declared by the SECTION directive in the source code. (If you used the **adds** command to create your symbols, as in case 4, the **di** command display shows NO.SECTION as the section name.)

The LOC (location) column of the **di** command display contains information that lets you correlate the display with your assembler listing. The symbols **START** and **ALOOP** in the **di** command display correspond to the labels **START** and **ALOOP** in the source code. When a location in the display does not correspond to a label in the symbol table, the **di** command substitutes the address of the instruction relative to the beginning of the section, as shown in the address field of your assembler listing.

If you haven't loaded your program's symbols and related information into the symbol table using a command such as **symlo**, the **di** command supplies only the absolute (actual) addresses in the LOC column. Since section **DEMO** begins at address 0, the relative address, or offset, is the same as the absolute address in this display. This offset feature is much more useful for sections that don't start at address 0.

Your system used the symbol table to translate numbers into symbols, making a display easier to read. Your system can also translate a symbol in a command line into an address. For example, because your system knows that the symbol **START** is equivalent to the address 100, you could have entered the **di** command in any of the following ways:

```
di 100 112
di START 112
di start start+12
di 100 START+12
di start,,10
```

A symbol can be entered in lowercase or uppercase.

The feature that enables DOS/50 and OS/40 to correlate symbols from your program with the numbers they represent is symbolic debug.

Put Values into the Table in Memory

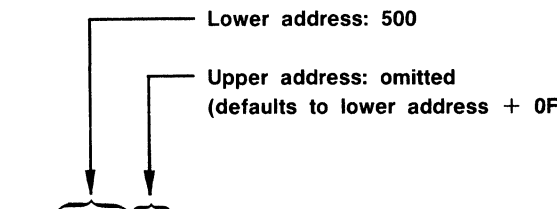
The demonstration program sums five numbers from a table in memory. Use the **p** (patch) command to store the numbers 1, 2, 3, 4, and 5 in the table. Remember that the symbol **TABLE** represents the table's address.

```
> p table 0102030405
```

Address of table: 500 String of bytes to be stored at addresses 500--504

Check the Contents of the Table

Use the **d** command to display the contents of the table. When you don't specify an upper boundary for the area to be displayed, the **d** command displays 16 bytes.



```

> d table
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000500 01 02 03 04 05 27 EB 8F C3 3C EB B6 9D 2B AB DB .....'<....<...+..
  
```

The data bytes from locations 500-504 (the table) contain the values you patched in. The data bytes from locations 505-50F contain the random data left over from previous system operations.

The following command displays only the contents of the table:

```

> d table table+tsize-1
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000500 01 02 03 04 05 .....
  
```

Check the Code Segment Register

The CS Register must be 0 for the demonstration run program to execute properly. Enter the following command to check the status of the CS Register:

```

> ds

PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000106        NMI=0 INTR=0 TEST=1    000F 0002 0000 0000 0000 0000
0000:0106      8MHZ                  0502 1007 0000 0000 0000 F006

      FLAGS . . . . OF DF IF TF  SF ZF . AF . PF . CF
      F006 X X X X  0 0 0 0  0 0 X 0  X 1 X 0
  
```

The CS Register is set to 0 when you select the 80186/80188 Emulator. If CS is not 0 (for example, if you have entered a RESET command), enter the following command to set CS to 0:

```

> s cs=0
  
```

Start Program Execution

Enter the **g** (go) command to start program execution at location **START**, the address specified by the **END** directive in the source code.

```
> g start
```

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+000112							
000112	NOP	000F	0000	0000	0000	0000	0000
0000:0113	90	0505	1007	0000	0000	0000	F006
000112	<BREAK						

The program executes. When the Exit SVC occurs, the program breaks, and the contents of the emulator registers are displayed. Symbol **AL** (lower-order byte of Register **A**) contains the sum of the numbers in the memory table: $1 + 2 + 3 + 4 + 5 = 0F$.

NOTE

*Register **A** (Symbol **AX** in the preceding display) is 000F. Symbol **AL** (lower-order byte of Register **A**) is 0F.*

MONITOR PROGRAM EXECUTION

You have assembled, loaded, and executed the demonstration program. The rest of this demonstration shows you some commands for monitoring program execution. These commands let you watch the changes in the emulator's registers and observe each instruction's effect as the program proceeds.

Trace All Instructions

The **tra** (trace) command lets you observe the changes in the 80186 Emulator registers as the program proceeds. When you enter a **tra** command and then start execution with the **g** command, display lines are sent to the system terminal. As each instruction executes, the display line shows the instruction (as in the **di** command display) and the contents of the registers after that instruction has executed. Enter the following command to trace all of the program's instructions:

```
> tra all
```

Enter the **g start** or **g 100** command line to resume program execution at the beginning of the program:

```
> g start
```

The following display shows the trace of each instruction as the program executes. Remember that you can type CTRL-S to suspend the display and CTRL-Q to resume the display.

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX,#0500	000F	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEMO+000103							
000103	MOVW CX,#0005	000F	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
DEMO+000106							
000106	XORB AL,AL	0000	0005	0000	0000	0000	0000
0000:0108	32C0	0500	1007	0000	0000	0000	F046
AL00P							
000108	ADDB AL,[BX]	0001	0005	0000	0000	0000	0000
0000:010A	0207	0500	1007	0000	0000	0000	F046
DEMO+00010A							
00010A	INC BX	0001	0005	0000	0000	0000	0000
0000:010B	43	0501	1007	0000	0000	0000	F002
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
DEMO+00010B							
00010B	LOOP \$-03	0001	0004	0000	0000	0000	0000
0000:0108	E2FB	0501	1007	0000	0000	0000	F002
AL00P							
000108	ADDB AL,[BX]	0003	0004	0000	0000	0000	0000
0000:010A	0207	0501	1007	0000	0000	0000	F046
DEMO+00010A							
00010A	INC BX	0003	0004	0000	0000	0000	0000
0000:010B	43	0502	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0003	0003	0000	0000	0000	0000
0000:0108	E2FB	0502	1007	0000	0000	0000	F002
AL00P							
000108	ADDB AL,[BX]	0006	0003	0000	0000	0000	0000
0000:010A	0207	0502	1007	0000	0000	0000	F006
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
DEMO+00010A							
00010A	INC BX	0006	0003	0000	0000	0000	0000
0000:010B	43	0503	1007	0000	0000	0000	F006

```

DEMO+00010B
00010B      LOOP      $-03      0006 0002 0000 0000 0000 0000
0000:0108      E2FB      0503 1007 0000 0000 0000 F006

AL00P
000108      ADDB      AL,[BX]    000A 0002 0000 0000 0000 0000
0000:010A      0207      0503 1007 0000 0000 0000 F002

DEMO+00010A
00010A      INC       BX        000A 0002 0000 0000 0000 0000
0000:010B      43        0504 1007 0000 0000 0000 F002

DEMO+00010B
00010B      LOOP      $-03      000A 0001 0000 0000 0000 0000
0000:0108      E2FB      0504 1007 0000 0000 0000 F002

PC/CS:IP  MNEMONIC/DATA      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
AL00P
000108      ADDB      AL,[BX]    000F 0001 0000 0000 0000 0000
0000:010A      0207      0504 1007 0000 0000 0000 F016

DEMO+00010A
00010A      INC       BX        000F 0001 0000 0000 0000 0000
0000:010B      43        0505 1007 0000 0000 0000 F006

DEMO+00010B
00010B      LOOP      $-03      000F 0000 0000 0000 0000 0000
0000:010D      E2FB      0505 1007 0000 0000 0000 F006

DEMO+00010D
00010D      MOVW      DX, 1007    000F 0000 0000 0000 0000 0000
0000:0110      BAO710      0505 1007 0000 0000 0000 F006

DEMO+000110
000110      OUT       DX,AL      000F 0000 0000 0000 0000 0000
0000:0111      EE          0505 1007 0000 0000 0000 F006

000110  <BREAK      TRACE>

```

After the accumulator is cleared, it begins to store the sum of the numbers being added. The `ADDB AL,[BX]` instruction adds a number from the table into the accumulator. At the end of the program, the accumulator contains the sum of the numbers you put into the table.

Register CX, the pass counter, is set to 5 (TSIZE) at the beginning of the program. It decreases by 1 because of the `LOOP` instruction each time a number is added into the accumulator. The program ends after Register CX reaches 0.

The BX Register, set to 500 (TABLE) at the start of the program, increases by 1 each time a number is added to the accumulator. At the end of the program, the BX Register has been incremented five times and contains 0505.

Trace to the Line Printer

By adding the parameter `\>LPT` to a command, you can direct that command's output to the line printer instead of the system terminal. First, verify that your line printer is properly connected and powered up. Then enter the following command to execute the program and direct the trace output to the line printer:

```
> g start \>LPT
```

If you're operating in TERM mode with an 8560, use one of the command lines shown in the following examples instead of the previous command line shown.

Example 1. To send the display to the 8560 line printer, enter:

```
> g start : lpr
```

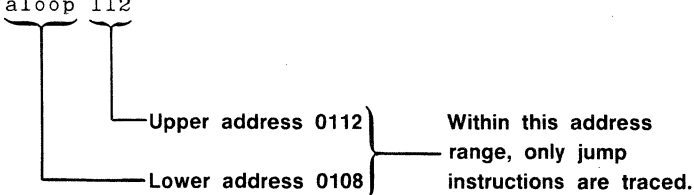
Example 2. To send the display to the 8540 or 8550 line printer, enter:

```
> g start \>LPT
```

Trace Jump Instructions Only

Another way to monitor the program's execution is to look only at the jump instructions. By tracing the jump instructions, you can still observe the changes in the registers, but you save time and space by not tracing the instructions within the loop. Enter the following command to trace only the jump instructions when the loop is being executed:

```
> tra jmp aloop 112
```



The **tra** command without parameters displays the trace conditions that are currently set. Because you can have up to three trace selections in effect at the same time, it is useful to see which selections are active. Check your trace status with the following command line:

```
> tra
TRACE    ALL,000000,0FFFFFFF
TRACE    JMP,AL00P,DEM0+000112
```

The preceding display shows that the **tra** command is set to trace all instructions for addresses 0-0108, to trace only jump instructions for addresses 0108-0112, and to trace all instructions for addresses 0113-FFFFF.

Start your program again with the **g** command. The following trace is displayed:

> g start

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX,#0500	000F	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEM0+000103							
000103	MOVW CX,#0005	000F	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
DEM0+000106							
000106	XORB AL,AL	0000	0005	0000	0000	0000	0000
0000:0108	32C0	0500	1007	0000	0000	0000	F046
DEM0+00010B							
00010B	LOOP \$-03	0001	0004	0000	0000	0000	0000
0000:0108	E2FB	0501	1007	0000	0000	0000	F002
DEM0+00010B							
00010B	LOOP \$-03	0003	0003	0000	0000	0000	0000
0000:0108	E2FB	0502	1007	0000	0000	0000	F002
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
DEM0+00010B							
00010B	LOOP \$-03	0006	0002	0000	0000	0000	0000
0000:0108	E2FB	0503	1007	0000	0000	0000	F006
DEM0+00010B							
00010B	LOOP \$-03	000A	0001	0000	0000	0000	0000
0000:0108	E2FB	0504	1007	0000	0000	0000	F002
DEM0+000110							
000110	OUT DX,AL	000F	0000	0000	0000	0000	0000
0000:0111	EE	0505	1007	0000	0000	0000	F006
000110	<BREAK TRACE>						

In the preceding display Register CX, the pass counter, is decremented; Register BX, the table pointer, is incremented; and AL, the accumulator, stores the sum of the numbers from the table. The contents of these registers are the same as the previous trace all display. With the trace jump instruction selection in effect, the instructions within the loop are not displayed.

Set a Breakpoint after a Specific Instruction

You have seen how the program adds the numbers together, now add only the third and fourth numbers from the table. To perform this task, you will want the pass counter to contain 2, and the table pointer to contain 502, the address of the third number in the table. You can make these changes without altering the object code in memory. First, stop program execution after the pass counter and the table pointer have been set. Next, while the program is stopped, enter new values for the pass counter and the table pointer. When execution resumes, the program will treat the new values as if they were the original programmed values.

Enter the following command line to trace all of the instructions as the program executes:

```
> tra all
```

Check the status of the trace with the following command line:

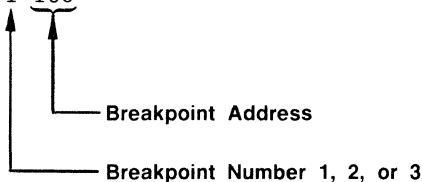
```
> tra
TRACE    ALL, DEMO+000000, 0FFFFF
```

The last **tra all** command makes earlier trace selections obsolete.

Set Breakpoints to Stop Program Execution

Now set a breakpoint to stop the program after the table pointer and the pass counter have been set. The following command causes the program to stop after it executes the **MOVW BX, #TABLE** instruction at address 103:

```
> bk 1 103
```



Use the **g** command to start program execution:

```
> g start
```

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX, #0500	000F	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEMO+000103							
000103	MOVW CX, #0005	000F	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
000103	<BREAK	TRACE, BKPT1>					

The **tra all** command displays all instructions up to and including the instruction at the breakpoint.

Set New Values in the Pass Counter and Table Pointer

Now that the program has reached the breakpoint, you can change the contents of the registers while execution is stopped. The displayed breakpoint shows that Register CX, the pass counter, contains 5, and Register BX, the table pointer, contains the address 500. Use the **s** (set) command to set the number of passes to 2 and set the table pointer to 502:

```
> s bx=502 cx=2
```

The **s** command does not produce a display, but you can use the **ds** (display status) command to check the values in the registers you changed. The **ds** command displays the contents of each emulator register and status flag. Check the result of the previous **s** command with the following command:

```
> ds
PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000106        NMI=0 INTR=0 TEST=1  000F 0002 0000 0000 0000 0000
0000:0106      8MHZ                0502 1007 0000 0000 0000 F006

      FLAGS      . . . .      OF DF IF TF      SF ZF      . AF      . PF      . CF
      F060      X X X X      0 0 0 0      0 0      X 0      X 1      X 0
```

The **ds** command shows that the pass counter and table pointer now contain the new values.

Resume Program Execution

If you enter the **g** command with no parameters, program execution starts where it left off.
Resume program execution after the breakpoint with the following command:

> g

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+000106							
000106	XORB AL,AL	0000	0002	0000	0000	0000	0000
0000:0108	32C0	0502	1007	0000	0000	0000	F046
AL00P							
000108	ADDB AL,[BX]	0003	0002	0000	0000	0000	0000
0000:010A	0207	0502	1007	0000	0000	0000	F006
DEMO+00010A							
00010A	INC BX	0003	0002	0000	0000	0000	0000
0000:010B	43	0503	1007	0000	0000	0000	F006
DEMO+00010B							
00010B	LOOP \$-03	0003	0001	0000	0000	0000	0000
0000:0108	E2FB	0503	1007	0000	0000	0000	F006
AL00P							
000108	ADDB AL,[BX]	0007	0001	0000	0000	0000	0000
0000:010A	0207	0503	1007	0000	0000	0000	F002
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
DEMO+00010A							
00010A	INC BX	0007	0001	0000	0000	0000	0000
0000:010B	43	0504	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0007	0000	0000	0000	0000	0000
0000:010D	E2FB	0504	1007	0000	0000	0000	F002
DEMO+00010D							
00010D	MOVW DX, 1007	0007	0000	0000	0000	0000	0000
0000:0110	BA0710	0504	1007	0000	0000	0000	F002
DEMO+000110							
000110	OUT DX,AL	0007	0000	0000	0000	0000	0000
0000:0111	EE	0504	1007	0000	0000	0000	F002
000110 <BREAK TRACE>							

The program performed two passes through the loop and added the third and fourth numbers in the table: $3 + 4 = 7$.

SUMMARY OF 80186 EMULATOR DEMONSTRATION RUN

You have assembled, loaded, executed, and monitored the demonstration run program. Review the commands you used:

sel	Selects the 80186/80188 Emulator
asm	Creates object code from an assembly language program
link	Links object code into a load module
f	Fills an area of memory with a specified value
d	Displays memory contents in ASCII and hexadecimal format
lo	Loads object code into memory
symlo	Loads program symbols for use in symbolic debug
di	Translates memory contents into assembly language mnemonics
p	Patches a string of bytes into memory
symd	Enables symbolic debug display
g	Begins or resumes program execution
tra	Selects instructions to be traced during program execution
bk	Sets a breakpoint
s	Modifies emulator registers
ds	Displays the contents of the emulator registers
adds	Adds symbols to the symbol table

DELETE THE DEMONSTRATION RUN FILES

Now that you have finished the demonstration run, you can delete the source file, object file, listing file, and load file. If you're using an 8550, the source file and load file are still available to you on the 8086/8088 Emulator installation disk. If you're using an 8560, once you delete the source file *asm* you cannot recover it.

Delete 8550 Files

If your files are on the 8550, use the following procedure to delete them.

First use the **user** command to move from the *DEMO* directory back into the directory */ROOT* you were in at the start of the demonstration. Recall that you marked that directory with the brief name

```
> user /ROOT
```

Now enter the following command to delete the *DEMO* directory and the files it contains:

```
> del DEMO/* DEMO
Delete ASM    ? y
Delete LOAD   ? y
Delete OBJ    ? y
Delete ASML   ? y
Delete DEMO   ? y
```

Before deleting each file, DOS/50 asks you whether you really want to delete the file. Type “y” for yes.

Delete 8560 Files

If your files are on the 8560, use the following procedure to delete them. Enter the following command to remove all files in the working directory, including the source file:

```
$ rm *
```

Now move from the *demo* directory back into the parent directory and remove the *demo* directory itself:

```
$ cd ..
$ rmdir demo
```

Turn Off Your System

For instructions on turning off your 8540 or 8550, refer to the Learning Guide of your System Users Manual.

Section 4

TECHNICAL INFORMATION

INTRODUCTION

This section contains technical reference material for the 80186/80188 Emulator and its associated prototype control probes. The technical reference material includes emulator timing relationships, the probe Power Supply Board calibration procedure, and specifications.

EMULATOR TIMING

The signals between the prototype and the emulating microprocessor are buffered. Therefore, some timing differences exist between the 80186/80188 Emulator and an 80186 or 80188 microprocessor inserted directly into the prototype.

Table 4-1 lists the emulator/microprocessor timing differences for the 80186/80188 Emulator. Figures 4-1, 4-2, 4-3, 4-4, and 4-5 contain timing diagrams corresponding to the signals listed in Table 4-1.

NOTE

The numbers in the left column of Table 4-1 are used to identify the signals in the timing diagrams.

Table 4-1
80186 Emulator Timing Differences

No.	Symbol	Parameter	Processor		Probe		Units
			Min.	Max.	Min.	Max.	
1	TDVCL	Data in Setup (A/D)	20		42		ns
2	TCLDX	Data in Hold (A/D)	10		0		ns
3	TARYHCH	Asynchronous Ready (AREADY) Active Setup Time ^a	20		42		ns
4	TARYLCL	AREADY Inactive Setup Time	35		57		ns
5	TCHARYX	AREADY Hold Time	15		0		ns
6	TSRYCL	Synchronous Ready (SREADY) Transition Setup Time	35		57		ns
7	TCLSRV	SREADY Transition Hold Time	15		0		ns
8	THVCL	HOLD Setup ^a	25		43		ns
9	TINVCH	INTR, NMI, TEST(L), TIMERIN, Setup ^a	25		41		ns
10	TINVCL	DRQ0, DRQ1, Setup ^a	25		41		ns
11	TCLAV	Address Valid Delay	10	44	10	46	ns
12	TCLAX	Address Hold	10		12		ns
13	TCLAZ	Address Float Delay	TCLAX	35	TCLAX-17	37	ns
14	TCHCZ	Command Lines Float Delay		45		71	ns
15	TCHCV	Command Lines Valid Delay (after float)		55		78	ns
16	TLHLL	ALE Width	TCLCL-35		1/2TCLCL-16		ns
17	TCHLH	ALE Active Delay		35		29	ns
18	TCHLL	ALE Inactive Delay		35		24	ns
19	TLLAX	Address Hold to ALE Inactive	TCHCL-25		TCHCL-37		ns
20	TCLDV	Data Valid Delay	10	44	0	46	ns
21	TCHDX	Data Hold Time	10		0		ns
22	TWHDX	Data Hold After WR(L)	TCLCL-40		TCLCL-51		ns
23	TCVCTV	Control Active Delay1	10	70	-4	52	ns
24	TCHCTV	Control Active Delay2	10	55	3	49	ns
25	TCVCTX	Control Inactive Delay	10	55	1	52	ns

^a To guarantee recognition at next clock.

Table 4-1 (Cont.)
80186 Emulator Timing Differences

No.	Symbol	Parameter	Processor		Probe		Units
			Min.	Max.	Min.	Max.	
26	TAZRL	Address Float to RD(L) Active	0		0		ns
27	TCLRL	RD(L) Active Delay	10	70	6	93	ns
28	TCLRHL	RD(L) Inactive Delay	10	55	8	50	ns
29	TRHAV	RD(L) Inactive to Address Active	TCLCL-40		TCLCL-67		ns
30	TCLHAV	HLDA Valid Delay	10	50	1	60	ns
31	TRLRH	RD(L) Width	2TCLCL-50		2TCLCL-85		ns
32	TWLWH	WR(L) Width	2TCLCL-40		2TCLCL-70		ns
33	TAVAL	Address Valid to ALE Low	TCLCH-25		TCLCH-44		ns
34	TCHSV	Status Active Delay	10	55	0	57	ns
35	TCLSH	Status Inactive Delay	10	55	0	57	ns
36	TCLTMV	Timer Output Delay		60		62	ns
37	TCLRO	Reset Delay		60		77	ns
38	TCHQSV	Queue Status Delay		35		37	ns
39	TCLCSV	Chip-Select Active Delay		66		68	ns
40	TCXCSX	Chip-Select Hold from Command Inactive	35		39		ns
41	TCHCSX	Chip-Select Inactive Delay	10	35	1	40	ns
42	TCKIN	CLKIN Period	62.5	250	62.5	250	ns
43	TCKHL	CLKIN Fall Time		10		10	ns
44	TCKLH	CLKIN Rise Time		10		10	ns
45	TCLCK	CLKIN Low Time	25		25		ns
46	TCHCK	CLKIN High Time	25		25		ns
47	TCICO	CLKIN of CLKOUT Skew		50		98	ns
48	TCLCL	CLKOUT Period	125	500	125	500	ns
49	TCLCH	CLKOUT Low Time	1/2TCLCL-7.5		1/2TCLCL-8.5		ns
50	TCHCL	CLKOUT High Time	1/2TCLCL-7.5		1/2TCLCL-6.5		ns
51	TCH1CH2	CLKOUT Rise Time		15		15	ns
52	TCL2CL1	CLKOUT Fall Time		15		15	ns

^a To guarantee recognition at next clock.

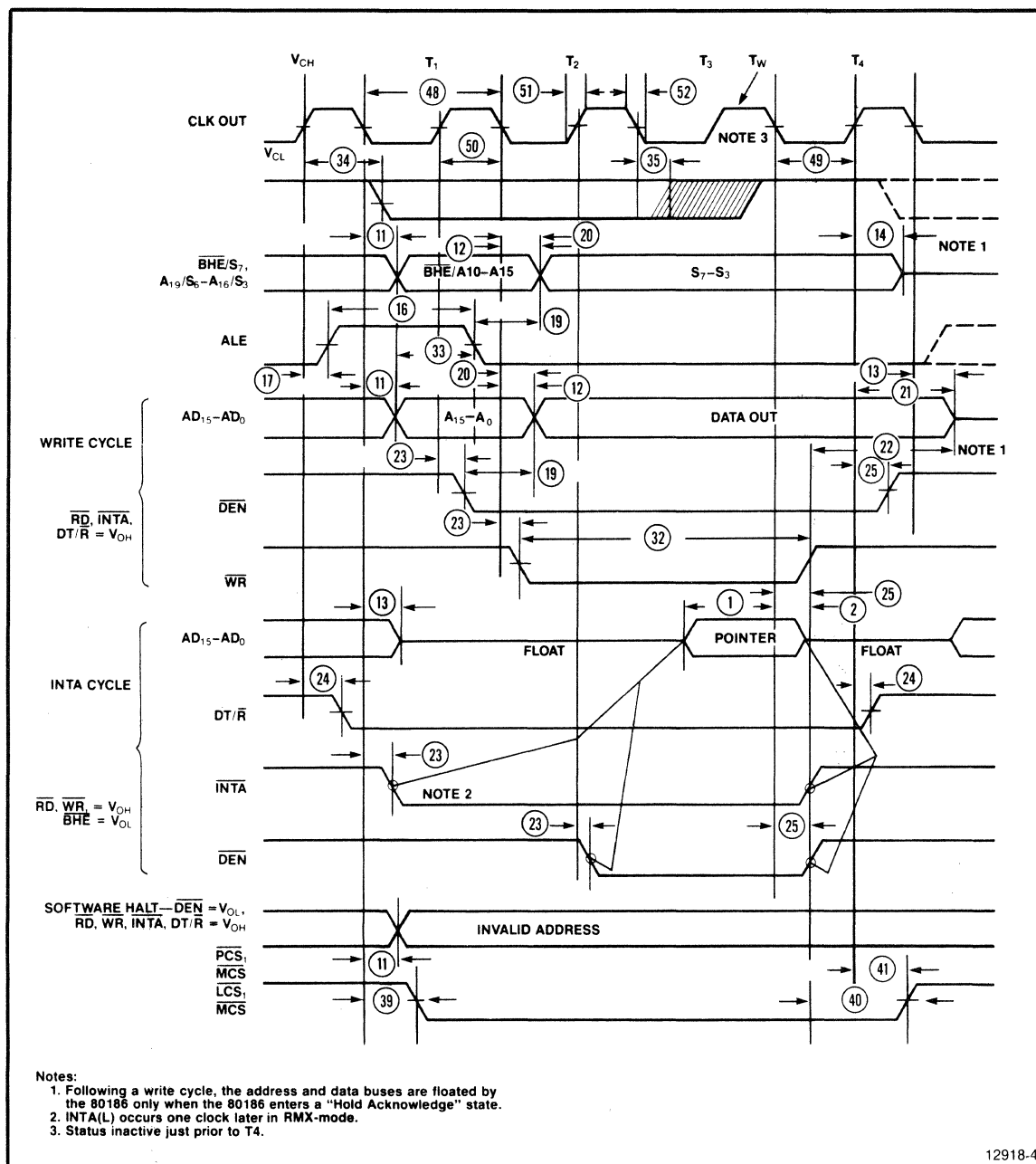


Figure 4-1. 80186 write cycle timing diagram.

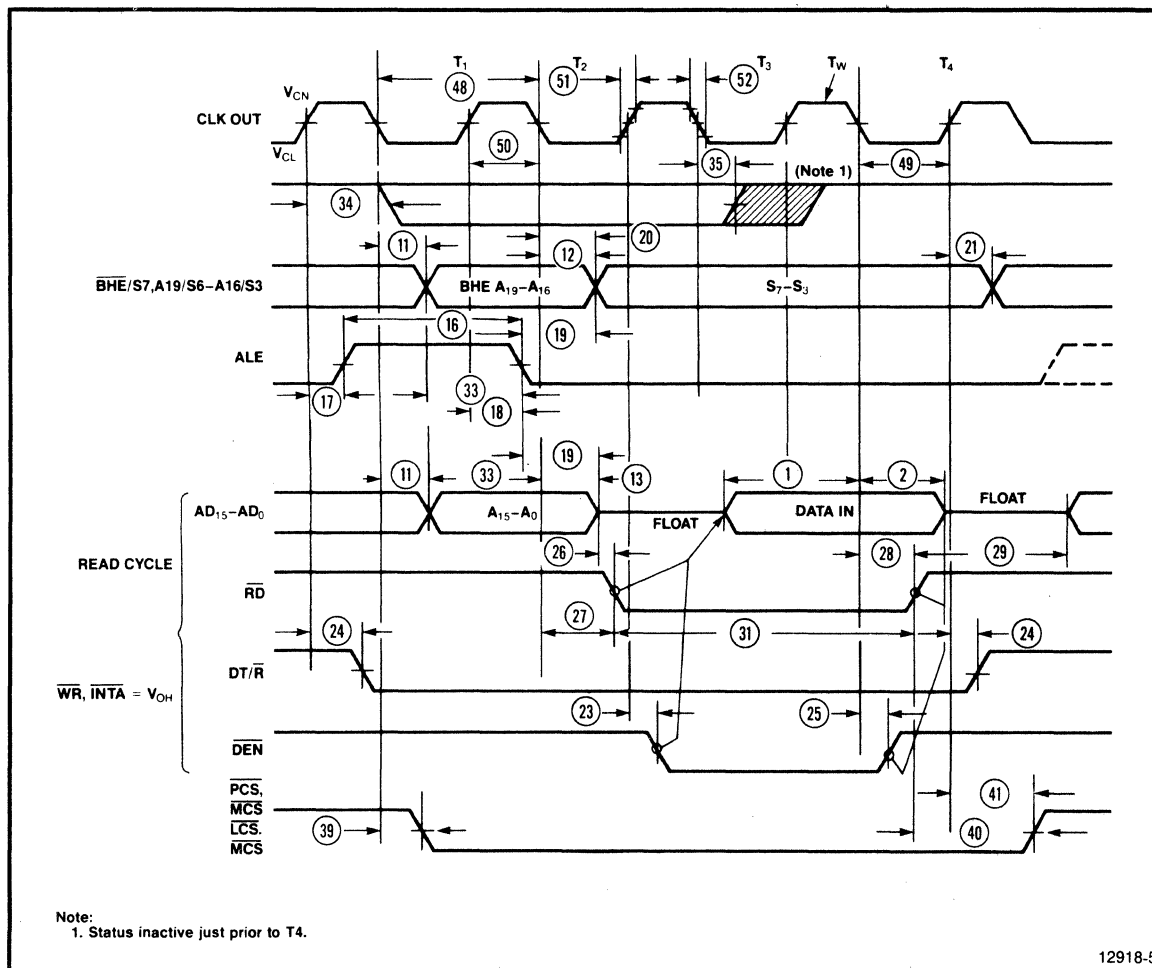


Figure 4-2. 80186 read cycle timing diagram.

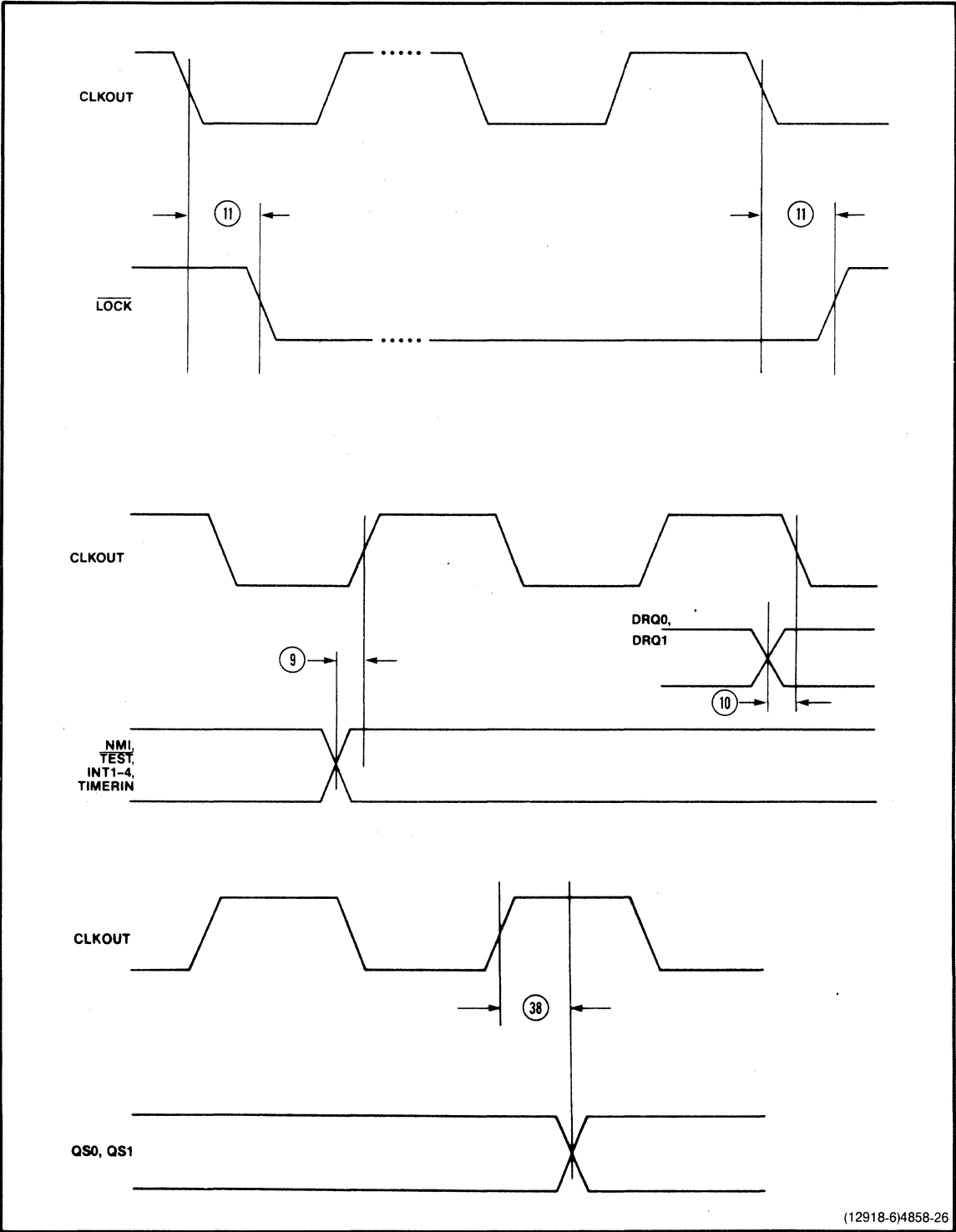


Figure 4-3. Clock timing diagram.

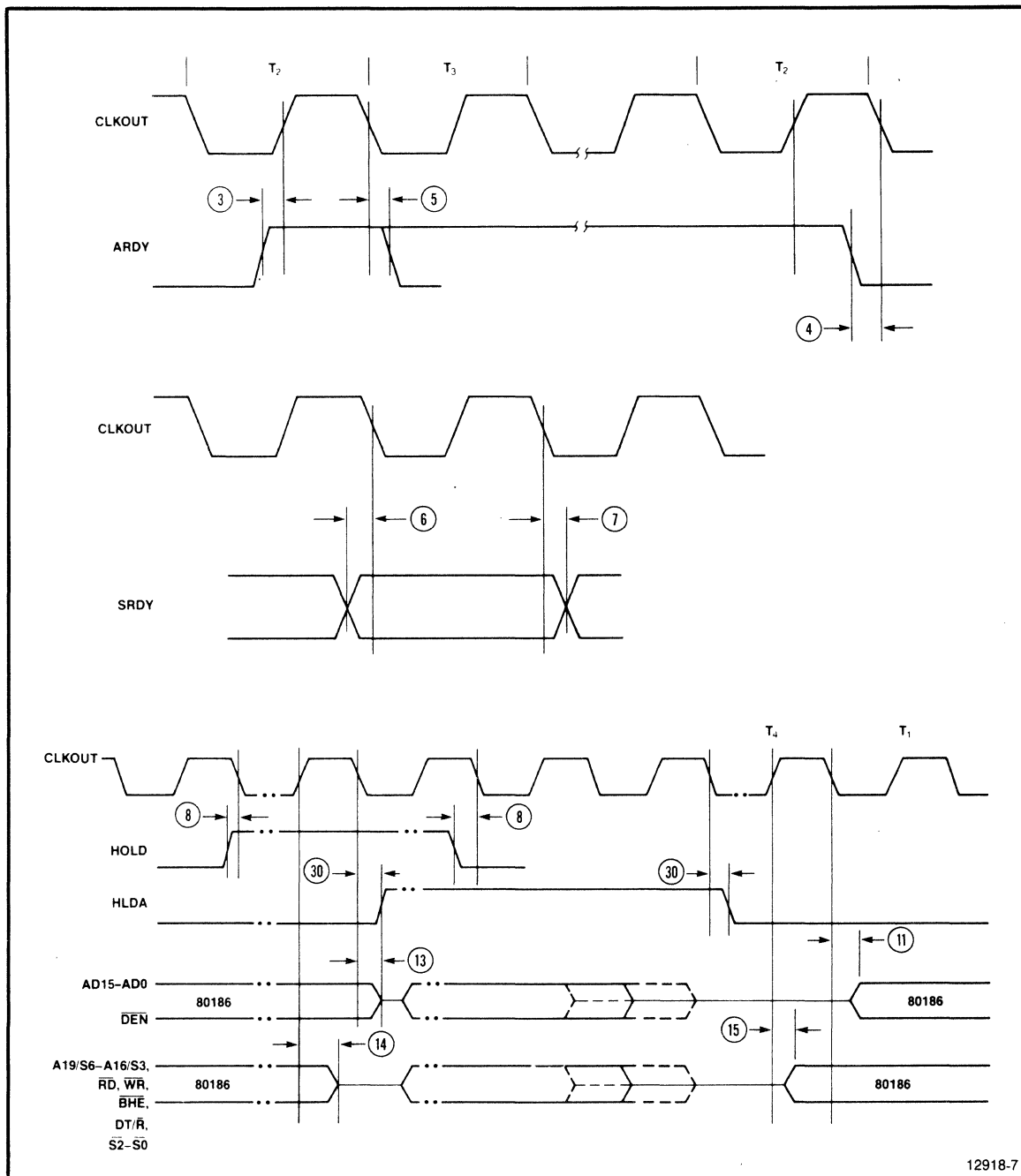


Figure 4-4. HOLD and HOLDA timing diagram.

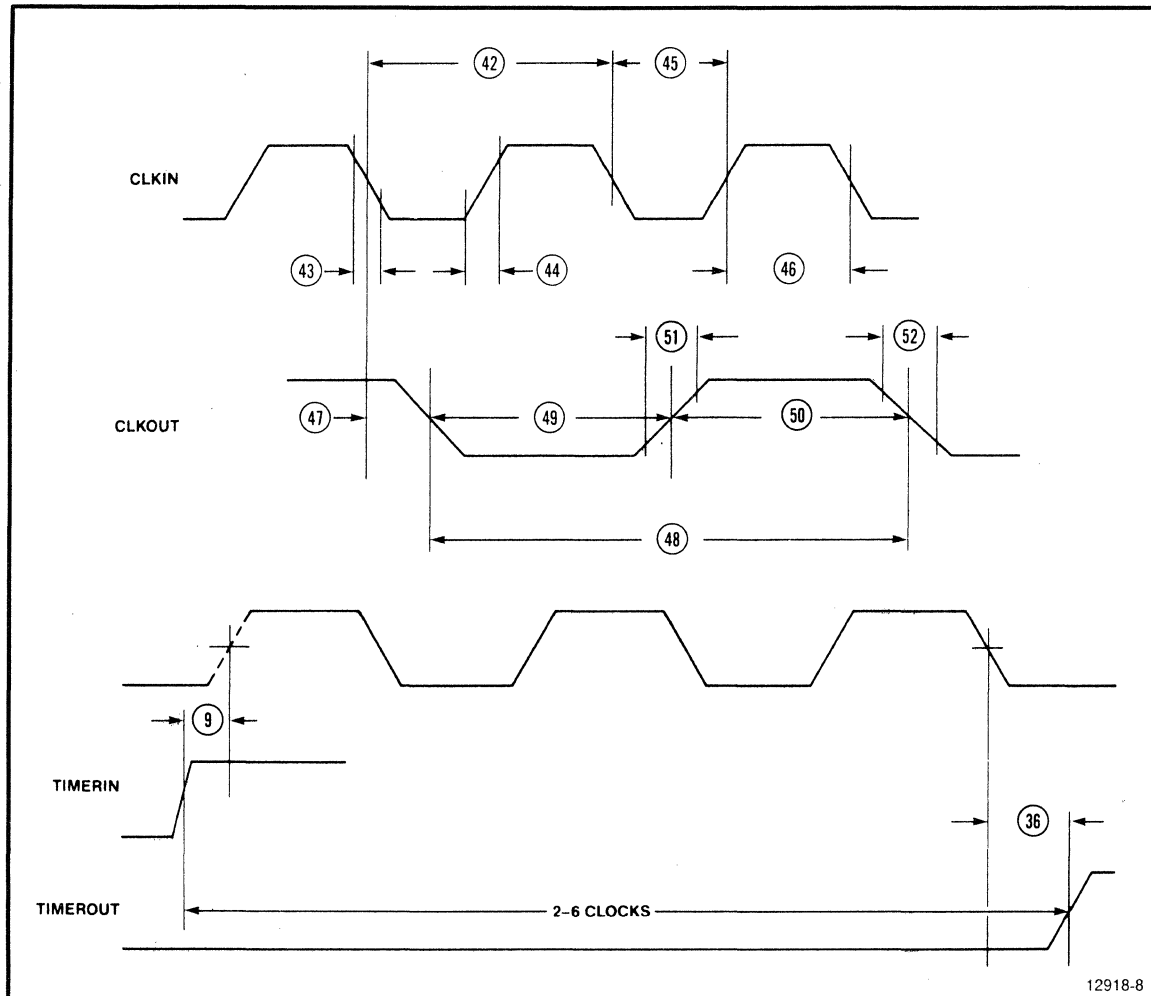


Figure 4-5. 80186 timer timing diagram.

POWER SUPPLY CALIBRATION

The power supply boards in both 80186 and 80188 prototype control probes are identical. The Power Supply Board generates two voltages required by the prototype control probe: +4.92 Vdc and +5.2 Vdc. The +4.92 Vdc supply provides power to part of the prototype control probe's Buffer Board. The +5.2 Vdc supply provides power to the 68-pin probe plug.

NOTE

You must disassemble the prototype control probe to calibrate the two power supply voltages. Before starting the calibration procedure, ensure that the power supply voltages require adjustment. Refer to the disassembly procedures for the prototype control probe contained in Section 6 of this manual under "Prototype Control Probe Disassembly Procedure".

The following text describes the calibration procedure used to adjust these two voltages. This procedure applies equally to the power supply boards in each probe.

Equipment Required

- TEKTRONIX 8550 Microcomputer Development Lab or 8540 Integration Unit, with 80186/80188 Emulator boards and 80186 or 80188 Prototype Control Probe installed.
- DVM with 100 μ V resolution and $\pm 0.1\%$ accuracy (TEKTRONIX DM501 or equivalent).

Procedure

1. Ensure that primary power (115 Vac or 230 Vac) to the development system is OFF.
2. Disassemble the prototype control probe so that the Power Supply Board is available to perform the calibration adjustments. Refer to the disassembly procedures in Section 6 of this manual under "Prototype Control Probe Disassembly Procedure".
3. Connect the DVM's negative test probe to one of the screws that holds the Power Supply Board to the top cover (ground).
4. Connect the DVM's positive test probe to TP1061 on the Power Supply Board. (See Figure 4-6).
5. Turn on the DVM and the microcomputer development system.
6. Enter the following command:

```
> sel 80186
```
7. Observe the voltage measured by the DVM.
8. Adjust R1051 on the Power Supply Board until the DVM reads +4.92 Vdc (voltage tolerance +4.90 to +4.94 Vdc).

NOTE

This adjustment is made with the power supply fully loaded (3.4 Amps). If the supply is not fully loaded the voltage could vary from +4.90 to +5.20 Vdc.

9. Disconnect the DVM's positive test probe from TP1061.
10. Connect the DVM's positive test probe to TP1011 on the Power Supply Board. (See Figure 4-6.)
11. Observe the voltage measured by the DVM.
12. Adjust R1011 on the Power Supply Board until the DVM reads +5.20 Vdc (voltage tolerance +5.00 to +5.25 Vdc).
13. Turn off power to all equipment.
14. Disconnect the DVM from the Power Supply Board.
15. Reassemble the prototype control probe interface assembly in reverse order of disassembly.

Section 5

JUMPERS

INTRODUCTION

This section defines the indicators, jumpers, and straps that are located on the emulator boards and the prototype control probe boards. The jumpers and straps enable you to configure your 80186/80188 Emulator to suit your prototype application. In addition, this section contains procedures to make changes to the jumpers on the development system's program memory boards.

EMULATOR BOARDS INDICATOR, JUMPERS, AND STRAP

The following text describes each indicator, jumper, and strap located on Board I and Board II. Board III has no indicators, jumpers, or straps.

NOTE

To access the jumpers on the emulator boards, you must remove the top cover from the development system and take the emulator boards from the card cage. The disassembly procedure is contained in Section 6 of this manual under the heading "Installing the Emulator Boards and Prototype Control Probe." Complete steps 1 through 3 of this procedure.

Board I Jumper and Strap

Board I has one jumper and one strap. Figure 5-1 shows the jumper and strap locations and their factory settings (default positions).

P1091

This jumper controls the Trigger Trace Analyzer (TTA) strobing during mode 3 operation. P1091 has two positions:

- | | |
|--------------|---|
| Pins 2 and 3 | TTA strobing is disabled during mode 3. This is the default position. |
| Pins 1 and 2 | TTA strobing is enabled during mode 3. (This position is used only as a diagnostic tool.) |

W4037

W4037 is a cuttable strap. The strap between pins 1 and 2 is cut at the factory and a strap is soldered between pins 2 and 3. A strap between pins 1 and 2 is not used.

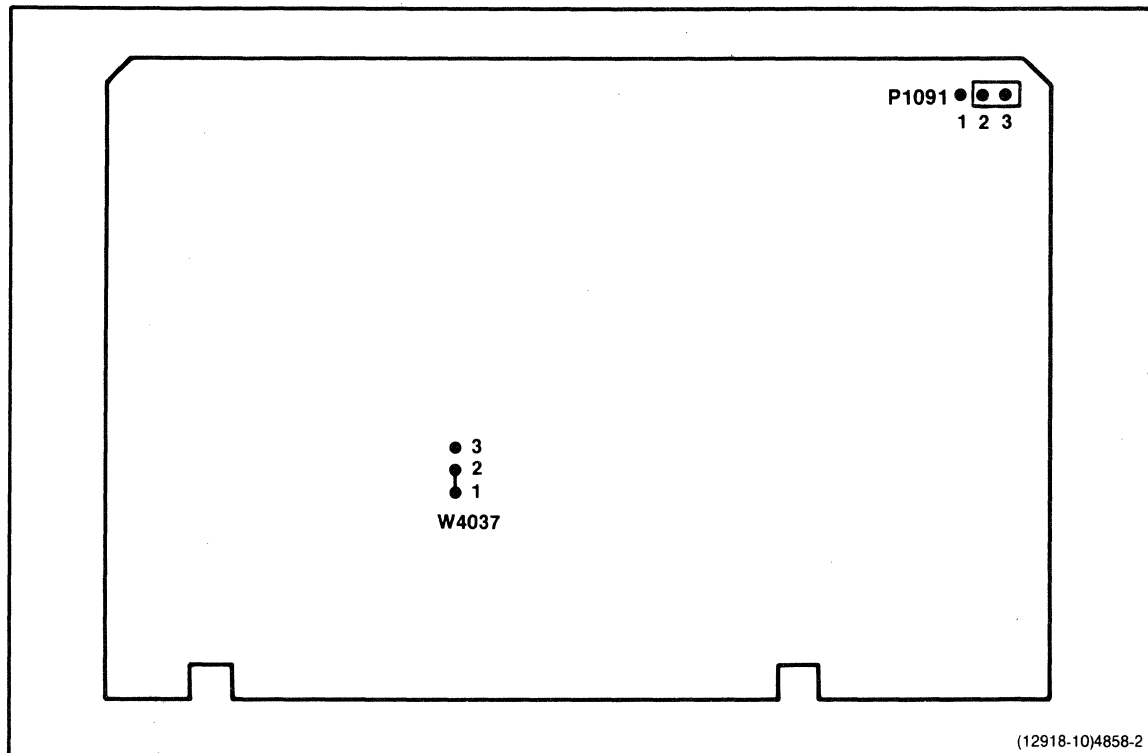


Figure 5-1. Board I jumper and strap locations.

Board II Indicator and Jumpers

Board II has one indicator and four jumpers. Figure 5-2 shows the indicator and jumper locations and the jumper factory settings (default positions).

DS3077

DS3077 is the Processor Halt indicator. This indicator is located near the center of Board II. When lit, this LED indicates that the emulating processor has halted.

P1011

This jumper controls a break option for write operations to protected areas of memory. P1011 has two positions:

- | | |
|--------------------|---|
| Pins 2 and 3 (YES) | The emulator breaks on a write to protected memory. This is the default position. |
| Pins 1 and 2 (NO) | The emulator does not break on a write to protected memory. |

P2089

This jumper controls a break option for read or write operations to nonexistent memory. P2089 is labeled MEMERR on the circuit board. P2089 has two positions:

- | | |
|-------------------|---|
| Pins 1 and 2 | The emulator breaks when nonexistent memory is accessed, and an error message is displayed. This is the default position. |
| Pins 2 and 3 (NO) | The emulator does not break when nonexistent memory is accessed. |

P6102

This jumper controls the wait state generation during program memory access. P6102 has three positions:

- | | |
|------------------|---|
| Pins 2 and 4 (D) | During program memory access, wait states are generated only for those locations with the al -s option. The number of wait states is determined by P7105's position. This is the default position. |
| Pins 1 and 2 (S) | During program memory access, wait states are generated for all locations. The number of wait states is determined by P7105's position. |
| Pins 2 and 3 (F) | During program memory access, no wait states are generated |

P7105

When a jumper block is placed across pins 2 and 4 or pins 1 and 2 of jumper P6102, jumper P7105 controls the number of wait states generated during program memory access. These wait states vary from one to seven.

P7105 consists of 14 pins. With the component side of the circuit board facing you and the 100-tab edge connector down, P7105 consists of seven two-pin positions numbered from 2 through 8.

Each two-pin position generates one less wait state than the corresponding position number. For example, if a jumper is placed across position 2, one wait state is generated. If a jumper is placed across position 8, seven wait states are generated.

P7105 is used when you want to duplicate user wait states in the development system. The **al** (allocate) command changes memory allocations to slow memory. P7105 is then positioned to generate the desired number of wait states.

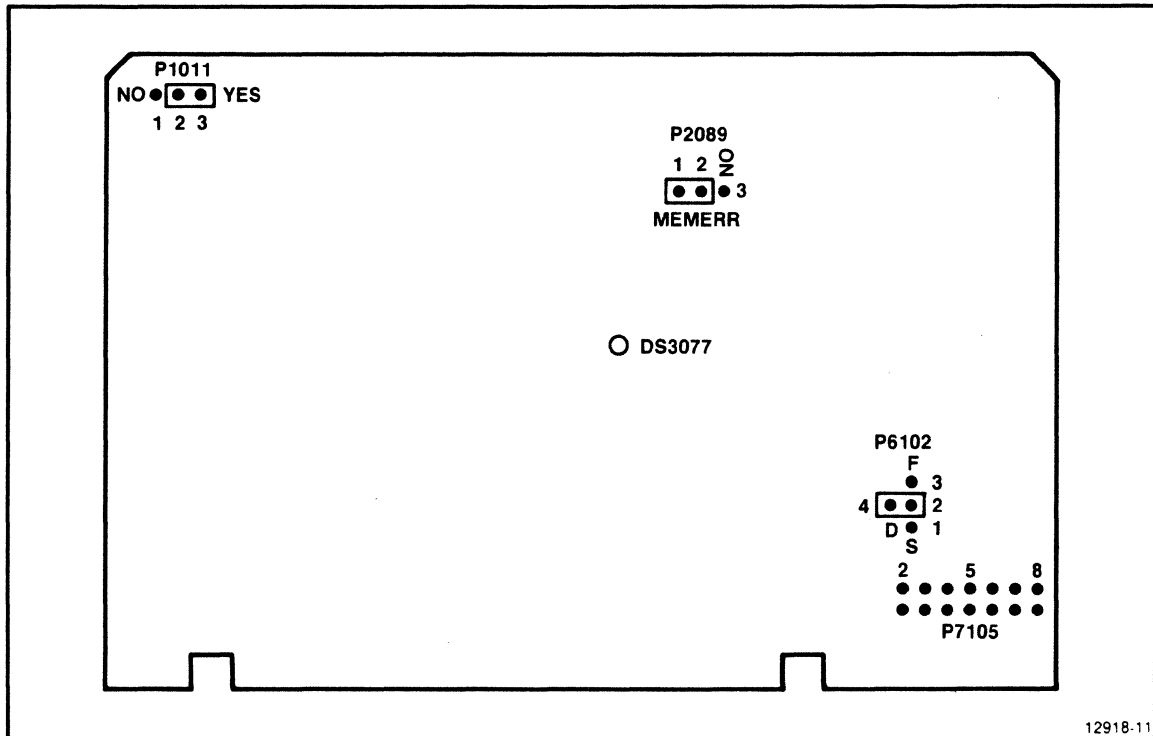


Figure 5-2. Board II jumper and indicator locations.

PROTOTYPE CONTROL PROBE INDICATORS AND JUMPERS

The following text describes each indicator and jumper located in the 80186/80188 Prototype Control Probe.

NOTE

To access the jumpers in the prototype control probe, you must disassemble the probe. To disassemble the probe housing, refer to Section 6 of this manual under the heading "Prototype Control Probe Disassembly Procedure".

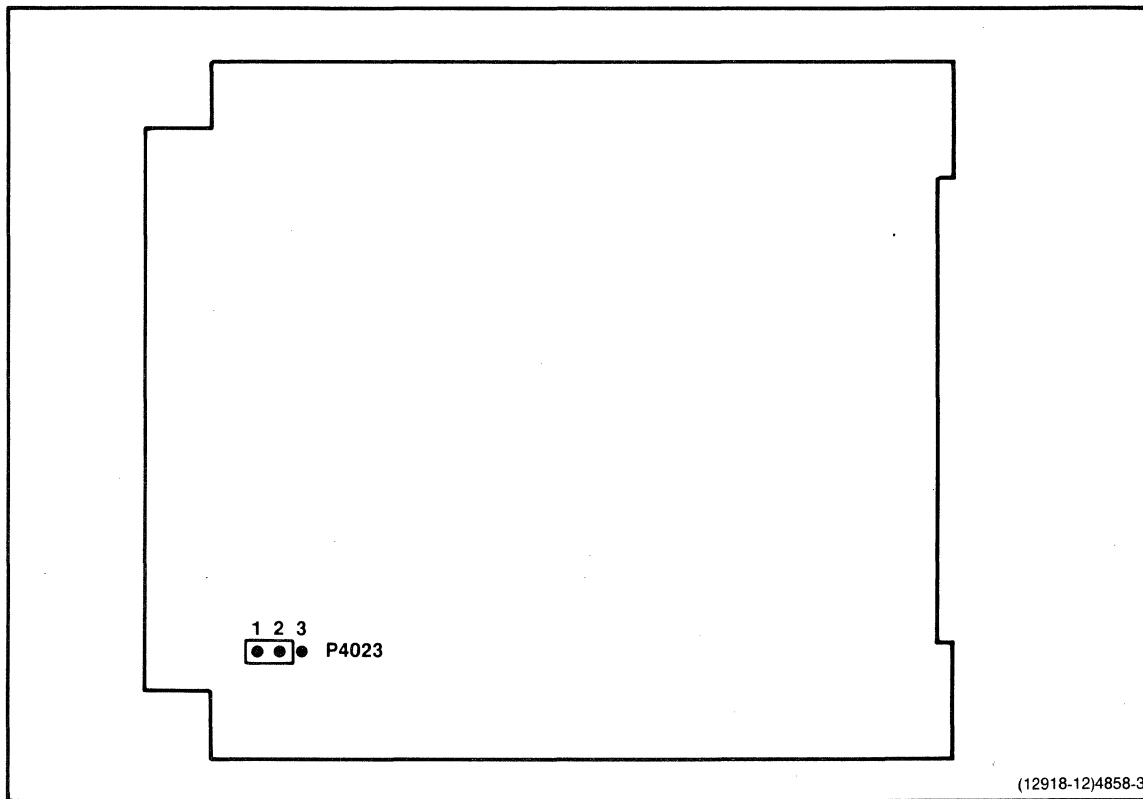
Control Board Jumpers

The Control Board has one jumper. This jumper's function is identical for both prototype control probes. Figure 5-3 shows the jumper location and its factory setting (default position).

P4023

This jumper controls the path from program or mapped-in memory to the prototype circuitry. The data path allows devices in a prototype, such as a co-processor, to follow program flow during emulation mode 1. P4023 has two positions:

- | | |
|--------------|--|
| Pins 1 and 2 | The data path is disabled. This is the default position. (Read operations from program memory are not seen by the prototype.) |
| Pins 2 and 3 | The data path is enabled during emulation mode 1. To avoid bus contention, buffers in the prototype circuitry are designed to support this option. |



(12918-12)4858-3

Figure 5-3. Control Board jumper locations.

Buffer Board Jumpers

The Buffer Board has eight jumpers. The functions of these jumpers are identical for both prototype control probes. Figure 5-4 shows the jumper locations and their factory settings. Figure 5-4 also shows the locations of three fuses. Refer to Section 4 of this manual for the type of fuses used.

P4023 and P6023

These jumpers select an option that allows the prototype's LSRDY(H) and LARDY(H) signals to control program memory access in emulation mode 1. P4023 affects SRDY control and P6023 affects ARDY control. In emulation mode 1, each program memory access causes a corresponding prototype memory access.

NOTE

Although data in prototype memory is not read, accessing this memory may cause the prototype to hang if the prototype is part of a multibus system.

P4023 is labeled SARDY-CNTRL on the circuit board. P4023 has two positions:

- | | |
|--------------|--|
| Pins 2 and 3 | The SRDY control option is disabled. This is the default position. |
| Pins 1 and 2 | The SRDY control option is enabled. |

P6023 is labeled ARDY-CNTRL on the circuit board. P6023 has two positions:

- | | |
|--------------|--|
| Pins 2 and 3 | The ARDY control option is disabled. This is the default position. |
| Pins 1 and 2 | The ARDY control option is enabled. |

P5023

P5023 is labeled BK-RDY-FLT on the circuit board. This jumper controls a READY fault break option. When selected, the option causes a break (with an error message) following a READY fault. P5023 is used with P7023 and P9011. Refer to Table 5-1 for information about the relationships between these three jumpers. P5023 has two positions:

- | | |
|--------------|--|
| Pins 1 and 2 | The break option is enabled. This is the default position. |
| Pins 2 and 3 | The break option is disabled. |

P7023

P7023 is labeled RDY-TOUT on the circuit board. This jumper controls a READY fault time-out option that forces READY true after a specified number of wait states (determined by P9011). P7023 is used with P5023 and P9011. Refer to Table 5-1 for information about the relationships between these three jumpers. P7023 has two positions:

- | | |
|--------------|--|
| Pins 1 and 2 | The time-out option is disabled. This is the default position. |
| Pins 2 and 3 | The time-out option is enabled. |

P8027

P8027 is labeled RES-SYNC on the circuit board. This jumper selects an option that can synchronize the reset signal from the prototype circuit with emulator bus cycles. When enabled, the option minimizes the chance of random writes to memory. P8027 has two positions:

- | | |
|--------------|---|
| Pins 1 and 2 | The reset synchronization option is disabled. This is the default position. |
| Pins 2 and 3 | The reset synchronization option is enabled. |

P9011

P9011 is labeled WAIT SEL on the circuit board. This jumper selects the number of wait states that precede a READY fault. P9011 is used with P5023 and P7023. Refer to Table 5-1 for information about the relationships between these three jumpers. P9011 has four positions:

- | | |
|--------------------|---|
| Pins 1 and 2 (NO) | This position selects zero wait states. This is the default position. |
| Pins 3 and 4 (1-2) | This position selects one or two wait states, depending on prototype timing. |
| Pins 5 and 6 (4-5) | This position selects four or five wait states, depending on prototype timing. |
| Pins 7 and 8 (8-9) | This position selects eight or nine wait states, depending on prototype timing. |

Table 5-1
Ready Fault/Time-out Jumpers

P9011	P5023	P7023	Function Produced
0 wait-states Pins 1-2	Pins 1-2 or 2-3	Pins 1-2 or 2-3	Fully transparent. The emulating processor waits indefinitely for one of the proto-type's ready lines USRDY(H) or UARDY(H) to go true. The positions of P5023's and P7023's jumper blocks have no effect when P9011's jumper block is across pins 1 and 2.
n wait-states Pins 3-4, 5-6, or 7-8	Pins 2-3	Pins 1-2	The emulating processor waits indefinitely for one of the prototype's ready lines USRDY(H) or UARDY(H) to go true. However, after "n" wait-states, any other break condition (such as CTRL-C) forces one of the emulating processor's ready lines LSRDY(H) or LARDY(H) true. A break is generated and and no error message is displayed.
	Pins 2-3	Pins 2-3	After "n" wait-states, the emulating processor times out and forces one of its ready lines LSRDY(H) or LARDY(H) true. A break is not generated and no error message is displayed.
	Pins 1-2	Pins 1-2 or 2-3	After "n" wait-states, the emulating processor times out and forces one of its ready lines LSRDY(H) or LARDY(H) true. A break is generated and an error message is displayed. The position of P7023's jumper block has no effect when P5023's jumper block is across pins 1 and 2.

n = The number of wait states generated by P9011's jumper block position.

P9013

This jumper controls a break option for user hold operations. P9013 has two positions:

- | | |
|--------------|---|
| Pins 2 and 3 | In this position, the emulator waits indefinitely for the current user hold cycle to end before any break request is honored. This is the default position. |
| Pins 1 and 2 | In this position, during a user hold operation any break request starts a timeout that lasts for about 0.5 second. At the end of this timeout the break request is honored. If P9013 is in this position (pins 1 and 2), the jumper block for P9015 must also be across pins 1 and 2. |

P9015

This jumper controls emulator-to-prototype hold (user hold) operations during internal operations (mode 3). P9015 has two positions.

- Pins 2 and 3** In this position, the user hold operations are enabled while the emulator is in mode 3. This position is selected if user hold operations are continuously required by the prototype circuitry (for example, refresh type functions). This is the default position.
- Pins 1 and 2** In this position, the user hold operations are disabled during mode 3. This position ensures stable memory data or status information during mode 3. This position is used with the dump or examine command to inspect memory affected by user hold operations.

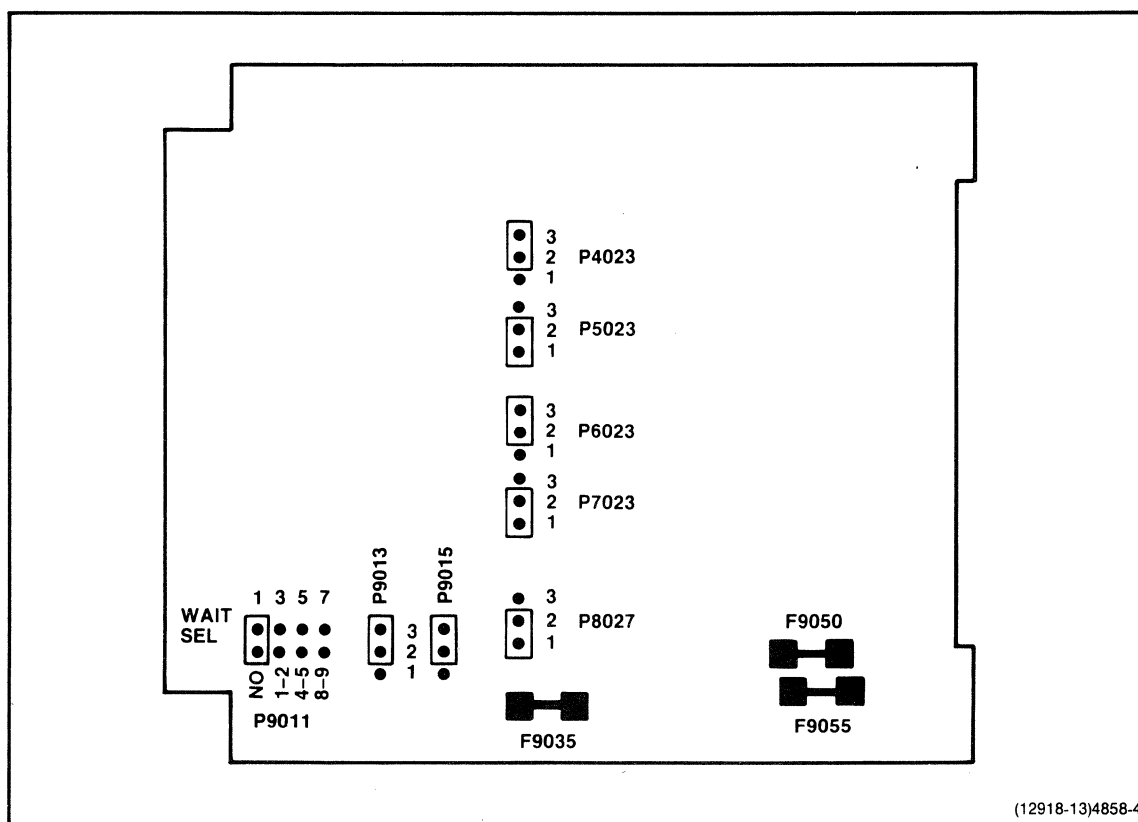


Figure 5-4. Buffer Board jumper and fuse locations.

Power Supply Board Indicators and Jumper

The Power Supply Board is used in the prototype control probes of several emulators. The Power Supply Board has seven indicators and one jumper.

There are six LED indicators that extend through the top of the prototype control probe's housing. The following list describes the functions of these six indicators:

RESET	When lit, DS6051 indicates the prototype has asserted the reset line URES(L).
TEST	When lit, DS5051 indicates the prototype has asserted the test line TEST LED(H).
WAIT	When lit, DS4051 indicates the prototype has the 80186 or 80188 microprocessor in a wait state.
HOLD ACK	When lit, DS3052 indicates the 80186 or 80188 microprocessor is in a hold state.
SELECT	When lit, DS3051 indicates the 80186 or 80188 Emulator is selected through the development system.
PROTOTYPE POWER	When lit, DS2051 indicates the prototype's power supply line is active.

Figure 5-5 shows the other indicator and jumper locations and the jumper's factory setting (default position).

DS1021

DS1021 is an overvoltage indicator. When lit, this LED indicates that the prototype control probe is shut down because of an overvoltage condition at the power supply's output.

P5061

This jumper's function is identical for both prototype control probes. P5061 has two positions:

Pins 1 and 2 (8086/8088)	This position configures the power supply for the 80186/80188 and 8086/8088 emulators.
Pins 2 and 3 (68000)	This position configures the power supply for the 68000 Emulator. This jumper position is not an option. The jumper exists only because the Power Supply Board is common to the prototype control probes of several emulators.

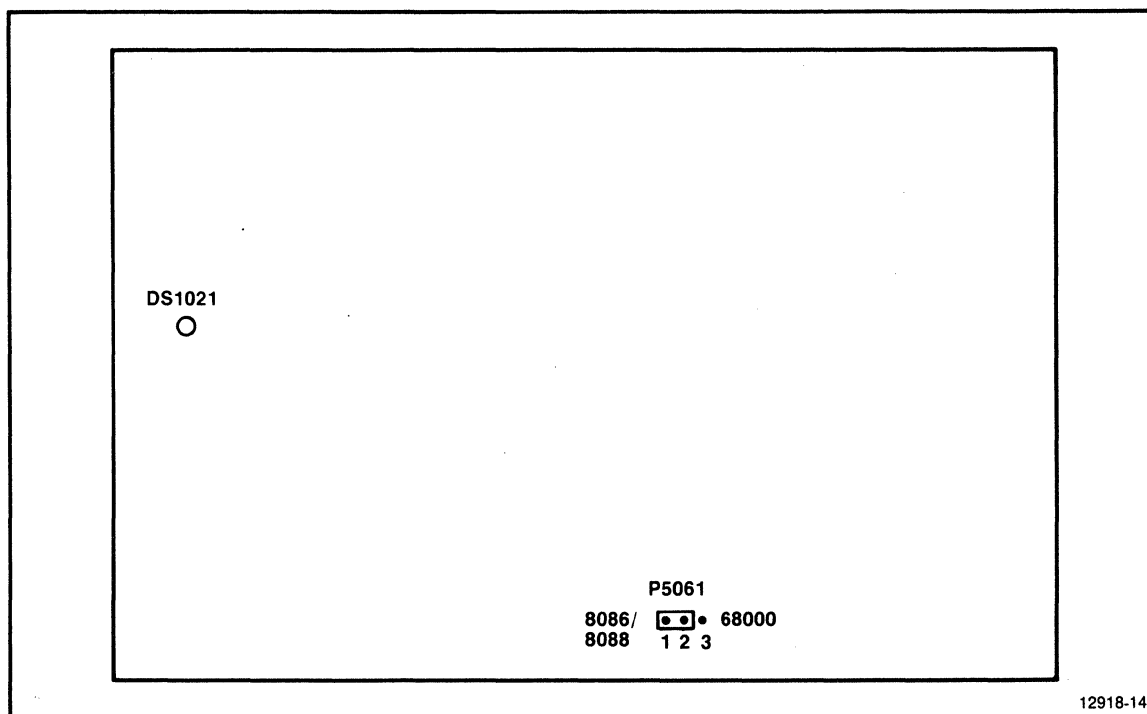


Figure 5-5. Power Supply Board jumper and indicator locations.

CPU Board Jumpers

The CPU Board has 16 jumpers. The functions of these jumpers are identical for both prototype control probes. Figure 5-6 shows the jumper locations and their factory settings (default positions).

P2067

This jumper controls the emulator interrupt type. P2067 has two positions:

- | | |
|--------------|---|
| Pins 1 and 2 | This position selects Nested Interrupt 3. This is the default position. |
| Pins 2 and 3 | This position selects Cascade INTRA1. |

P2068

This jumper controls the emulator interrupt type. P2068 has two positions:

Pins 1 and 2 This position selects Nested Interrupt 2. This is the default position.

Pins 2 and 3 This position selects Cascade INTRA0.

NOTE

During RMX operation, P2067 and P2068 should be in the cascade position. Place the jumper blocks for both P2067 and P2068 across pins 2 and 3.

Dummy Address Jumpers

The Dummy Address jumpers select a block of 256 “dummy” addresses to which pseudocode fetches can be directed. These addresses are required because a system access to prototype memory or I/O may cause superfluous read cycles. The dummy addresses must not include any sequential memory-mapped I/O devices or memory used for another purpose.

There are 12 dummy address jumpers (DA8—DA19). Each jumper corresponds to one of the twelve most significant address lines. Once the jumpers select a dummy address block, any superfluous read cycles are directed to addresses in that block. The individual address that is accessed will be determined by the eight least significant address lines (A0–A7).

Each dummy address jumper has three pins. If a jumper block is placed across pins 1 and 2 (default), the address line associated with that jumper is low. If a jumper is placed across pins 2 and 3, the address line is held high. Table 5-2 shows the address lines associated with the 12 dummy address jumpers.

Table 5-2
Dummy Address Jumpers

Address Line	Jumper	Address Line	Jumper
A8	P2019	A14	P2013
A9	P2018	A15	P2012
A10	P2017	A16	P4044
A11	P2016	A17	P4043
A12	P2015	A18	P4042
A13	P2014	A19	P4041

P6042

This jumper identifies the emulating processor. P6042 has two positions:

- | | |
|--------------|--|
| Pins 1 and 2 | This position is used for an 80186 microprocessor. |
| Pins 2 and 3 | This position is used for an 80188 microprocessor. |

P6043

This jumper selects an input clock frequency of 8 or 16 MHz for the emulating processor in mode 0. P6043 has two positions:

- | | |
|--------------|---|
| Pins 1 and 2 | This position selects an input clock frequency of 16 MHz. |
| Pins 2 and 3 | This position selects an input clock frequency of 8 MHz. |

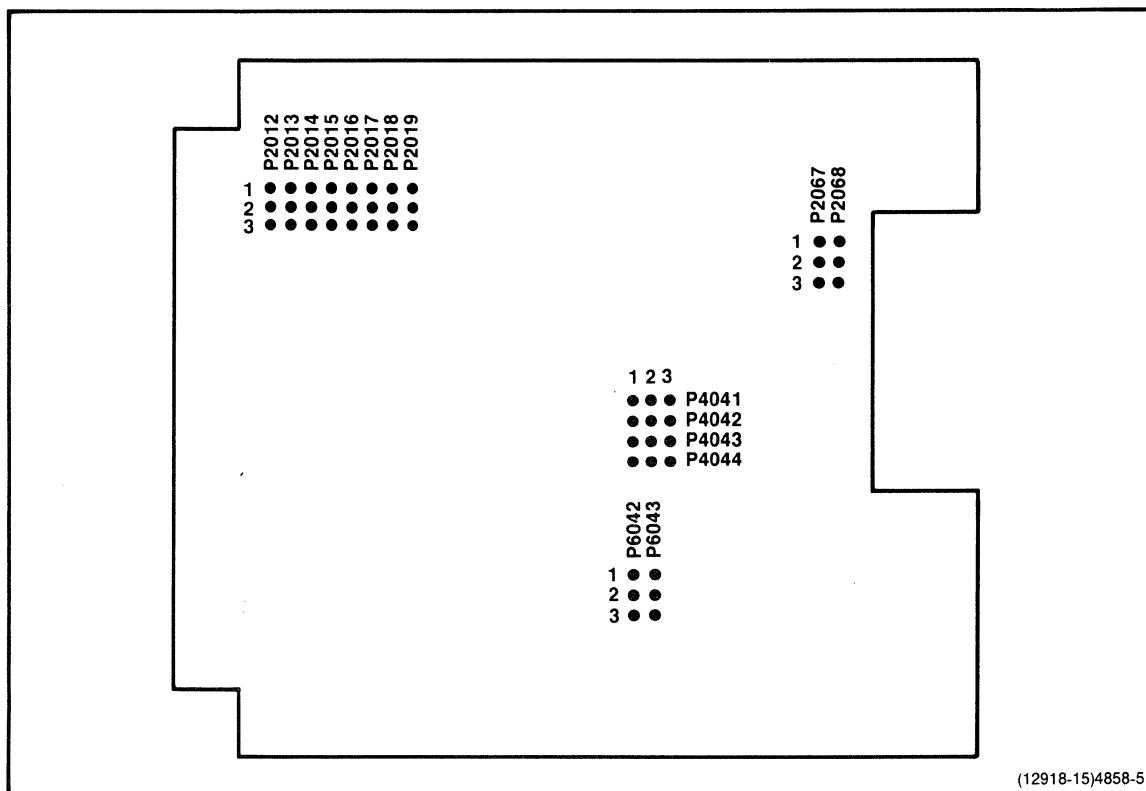


Figure 5-6. CPU Board jumper locations.

DEVELOPMENT SYSTEM JUMPERS AND STRAPS

When the 80186/80188 Emulator is installed in an 8500 Series development system, each 32K Program Memory Board or 64/128 Program Memory Board in the development system must be configured to support the specific emulator's operation. This requires that several jumpers be checked for proper positioning.

NOTE

To access jumpers on the memory boards, you must remove the top cover from the development system and take the memory boards out of the card cage. The disassembly procedure is contained in Section 6 of this manual under the heading "Installing the Emulator Boards and Prototype Control Probe". Complete steps 1 through 3 of this procedure.

32K Program Memory Board Jumper Adjustments

The following text tells you how to modify the development system's 32K Program Memory Board to provide 80186/80188 Emulator support. Refer to your development system's installation manual for information about the location and function of the strap, switch, and jumpers.

- W5011 Delayed read strap. Cut the ECB run between pins 1 and 2, then solder a strap between pins 2 and 3.
- S7170 Extended memory DIP switch. All switches on this DIP switch must be set to 0 (the ON position).
- J6175 Low/High board jumper. If there is only one 32K memory board in the development system's Program Section, place the jumper block across pins 2 and 3 of J6175. If there are two 32K memory boards, place the jumper blocks across pins 1 and 2 on one board and across pins 2 and 3 on the other board.
- J7171 Extended bank jumper. This jumper block must be placed across pins 1 and 2.

64K/128K Program Memory Board Jumper Adjustments

The following text tells you how to set the jumpers on a 64K/128K Program Memory Board to provide 80186/80188 Emulator support. Refer to your development system's installation manual for information about the location and function of these jumpers.

- J7090 Read delay jumper. This jumper block must be set to the 0 ns position (no delay).
- J8100 through J8160 Address select jumpers. These jumpers must be set to select continuous memory from 0 to 128K. This is done by setting jumper blocks of group A to 0, of group B to 32K, of group C to 64K, and of group D to 96K. Refer to the *64K/128K Static Program Memory Installation Manual* for more information about setting these jumpers.

80186/80188 AND DEVELOPMENT SYSTEM JUMPER SUMMARY

Table 5-3 is a jumper summary for the 80186/80188 Emulator showing the jumper default positions. Table 5-4 is a jumper summary for the memory boards in the development system.

Table 5-3
80186/80188 Jumper Default Position Summary

Circuit Board	Jumper	Default position
Board I	P1091	Jumper across pins 2-3
Board II	P1011 P2089 P6102 P7105	Jumper across pins 2-3 Jumper across pins 1-2 Jumper across pins 2-4 No default; refer to description
Control Board	P4023	Jumper across pins 1-2
Buffer Board	P4023 P5023 P6023 P7023 P8027 P9011 P9013 P9015	Jumper across pins 2-3 Jumper across pins 1-2 Jumper across pins 2-3 Jumper across pins 1-2 Jumper across pins 1-2 Jumper across pins 1-2 Jumper across pins 2-3 Jumper across pins 2-3
CPU Board	P2067 P2068 P2012 P2013 P2014 P2015 P2016 P2017 P2018 P2019 P4041 P4042 P4043 P4044 P6042 P6043	Jumper across pins 1-2 Jumper across pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 Dummy address jumper, pins 1-2 No default; refer to description No default; refer to description
Power Supply Board	P5061	Jumper across pins 1-2

Table 5-4
System Jumper Default Position Summary

Circuit Board	Jumper	Default Position
32K Memory Board	W5011	Cut the ECB run between pins 1 and 2, then solder a strap between pins 2 and 3.
	S7170	Set each position of this DIP switch to 0 (the ON position).
	J6175	Place jumper across pins 1 and 2 if if have only one board. If you have two boards, place jumper across pins 1 and 2 on one board and across pins 2 and 3 on the other board.
	J7171	Place jumper across pins 1 and 2.
64K/128K Memory Board	J7090	Set to 0 ns position (no delay).
	J8100-J8160	Set jumper group A to 0, group B to 32K, group C to 64K, and group D to 96K.

Section 6

INSTALLATION

INTRODUCTION

This section tells how to install the 80186/80188 Emulator boards and prototype control probe in your Tektronix development system. This section contains information for connecting the 68-pin probe plug to your prototype circuit, and disassembly instructions for the prototype control probe. This section also describes how to install the control software for the various development systems.

HARDWARE INSTALLATION



*Before inserting or removing any circuit board, ensure that primary power to the development system is **OFF**. Inserting or removing a board while the power is **ON** may result in component damage.*

Under no circumstances can another emulator be installed in your Tektronix development system while the 80186/80188 Emulator is installed. Excessive power supply loading will result.

NOTE

The 80186/80188 Emulator and the Memory Allocation Controller Board cannot be installed in the same development system.

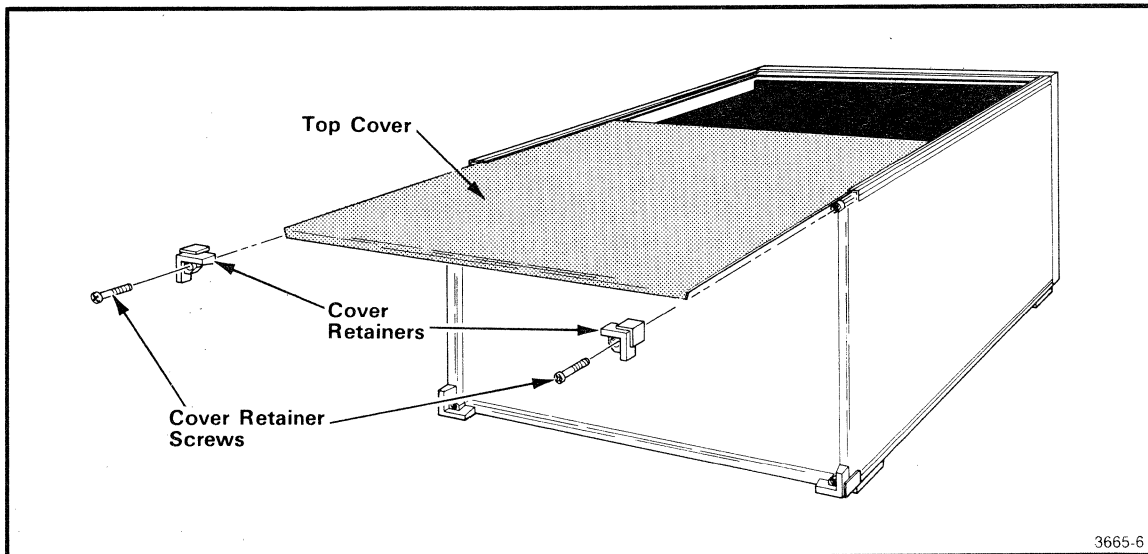


Figure 6-1. Removal/installation of the top cover.

Installing the Emulator Boards and Prototype Control Probe

The 80186 or 80188 Prototype Control Probe attaches to the 80186/80188 Emulator boards by two 6-foot ribbon cables at connectors P100 and P200 on Board II. To install the 80186/80188 Emulator, perform the following procedure.

1. Verify that primary power (115 Vac or 230 Vac) to the development system is off.
2. Remove the two cover retainers at the upper rear corners of the mainframe. Figure 6-1 illustrates the top cover removal.
3. Remove the top cover by sliding it straight back, then set the cover aside.

NOTE

Several development system boards (32K Memory Board and 64/128K Memory Board) have jumpers that must be changed when the 80186/80188 Emulator is installed. If you have these boards installed in your system, check to make sure these jumpers are correctly set before installing the 80186/80188 Emulator boards. Section 5 of this manual describes the correct settings for these jumpers.

4. Remove any emulator board or Memory Allocation Controller Board that is installed in the development system's card cage. Refer to the Caution and Note paragraphs at the beginning of this section before removing any boards.
5. Connect Boards I, II, and III together, using the six 50-conductor ribbon cables provided. Dress the cables as illustrated in Figure 6-2.

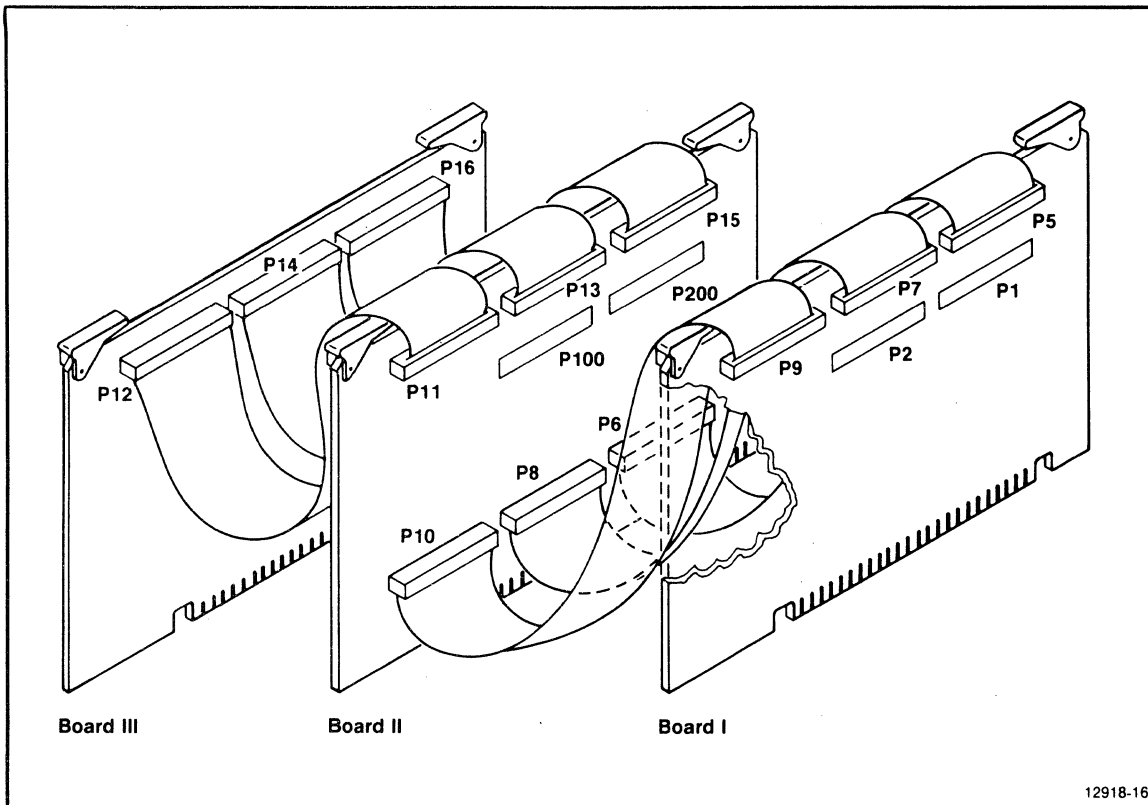


Figure 6-2. 80186/80188 Emulator board interconnections.

6. Face the front of the mainframe and hold the three emulator boards by their upper side edges. With the component sides facing left, align the boards with the other circuit boards in the development system's card cage.
7. While holding all three boards, guide Board I into the vertical channel of J14 on the Main Interconnect Board. Next, guide Board II into the vertical channel of J15, then guide Board III into the vertical channel of J16. Refer to Figure 6-3 or Figure 6-4 for the recommended board arrangements of your development system.

NOTE

When the emulator boards reach their connectors on the Main Interconnect Board, do not snap them into place yet. You may need to lift the boards slightly when performing later steps in this procedure.

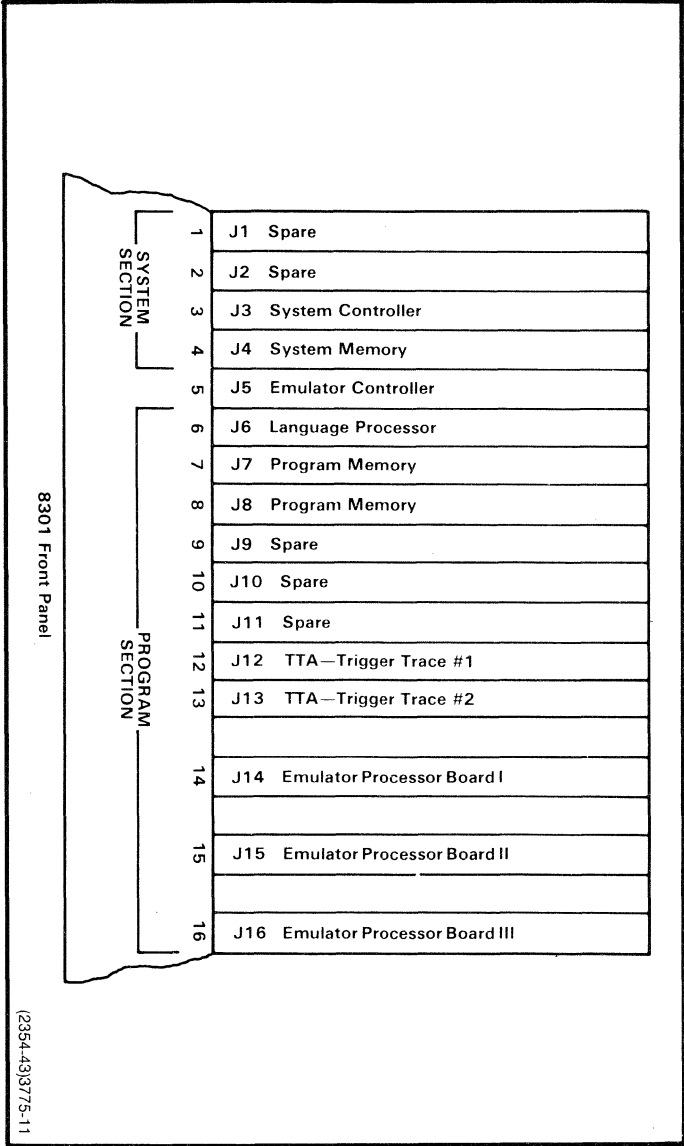


Figure 6-3. Recommended board arrangement for the 8301.

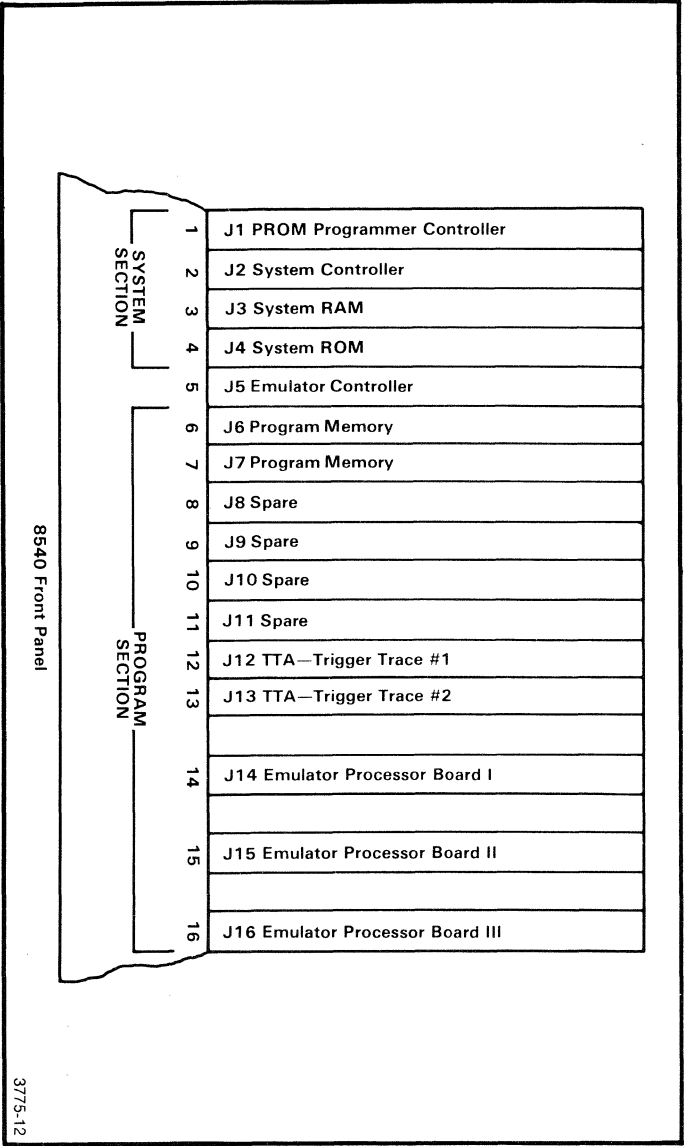


Figure 6-4. Recommended board arrangement for the 8540.

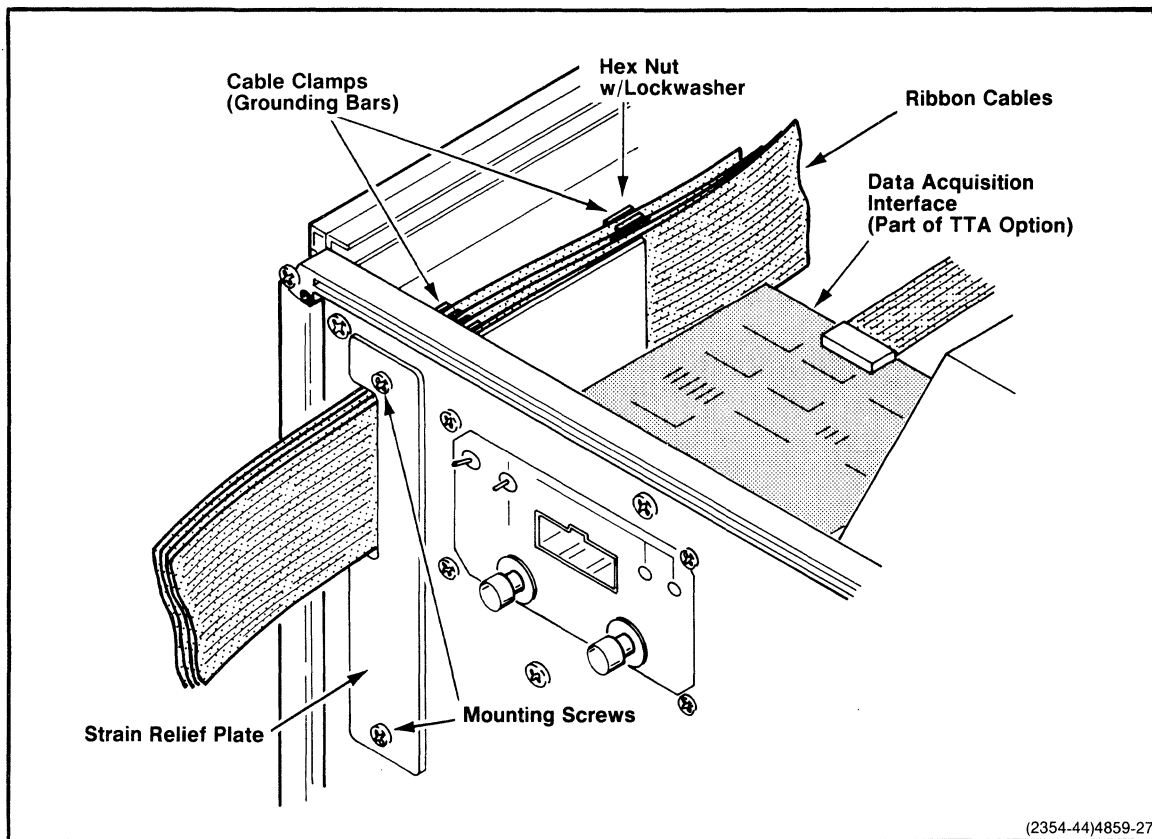


Figure 6-5. Strain relief plate installation/removal.

8. Remove the two mounting screws at the top and bottom of the strain relief plate, as shown in Figure 6-5. Then remove the strain relief/cable clamp assembly from the rear panel.

NOTE

The standard cable clamps provided with your development system are not compatible with the 80186/80188 Prototype Control Probe ribbon cables. The standard clamps must be replaced with the clamps provided with your 80186 or 80188 Prototype Control Probe.

9. Mount the two 6-foot prototype control probe ribbon cables to the cable clamp assembly using the new clamps provided. Secure the cable clamps with hex nuts and lock washers. Figure 6-6 illustrates the ribbon cable installation. Allow enough ribbon cable to reach connectors P100 and P200 on Board II in J15 of the Main Interconnect Board in the development system's card cage.
10. Feed the prototype control probe cable connectors J100 and J200 through the cableway opening. Then guide the strain relief/cable clamp assembly into the cableway opening. Attach the strain relief plate to the rear panel using the two screws removed in step 8.

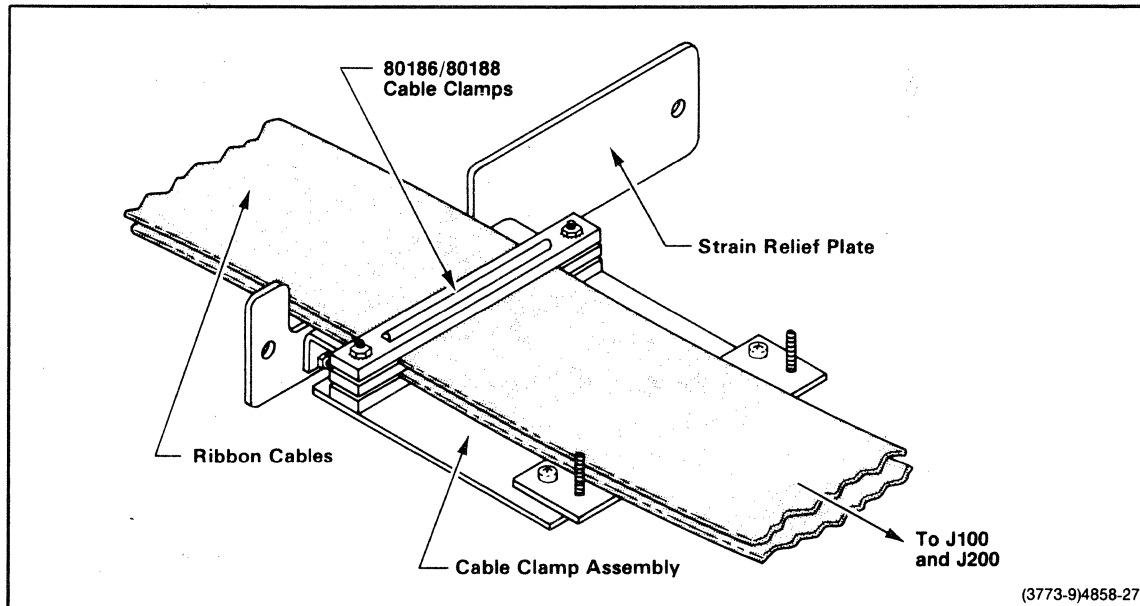


Figure 6-6. Ribbon cable installation.

11. Attach the two ribbon cable connectors labeled J100 and J200 to P100 and P200 on Board II, as shown in Figure 6-7. Be sure that pin 1 of each cable connector (blue stripe) and pin 1 of each socket are aligned. Pin 1 of P100 and P200 is to the left when viewed from the component side of the board.

NOTE

If you are installing your emulator in an 8540 and have not yet installed the option ROMs shipped with your emulator, refer to "Installing the 8540 Firmware," later in this section.

If you intend to install, or have already installed, the optional Trigger Trace Analyzer in your development system, refer to "Connecting to the Trigger Trace Analyzer (Optional)," later in this section. Complete the connections to the Trigger Trace Analyzer before proceeding.

12. Slide the three emulator boards down until they reach their respective connectors on the Main Interconnect Board in your development system's mainframe. Then press down firmly and evenly on the top edges, one board at a time, until each board snaps into place.
13. Dress the cables along the cableway at the side of the mainframe and over the side of the card cage. Make sure the cables are dressed to lie flat and allow clearance for the top cover.
14. Slide the top cover back into the guide tracks at the top of the mainframe. Be sure the cover is properly seated in the slot at the front of the mainframe.
15. Install the cover retainers at the upper corners on the rear of the mainframe, as shown in Figure 6-1. Tighten the cover retainer screws securely.

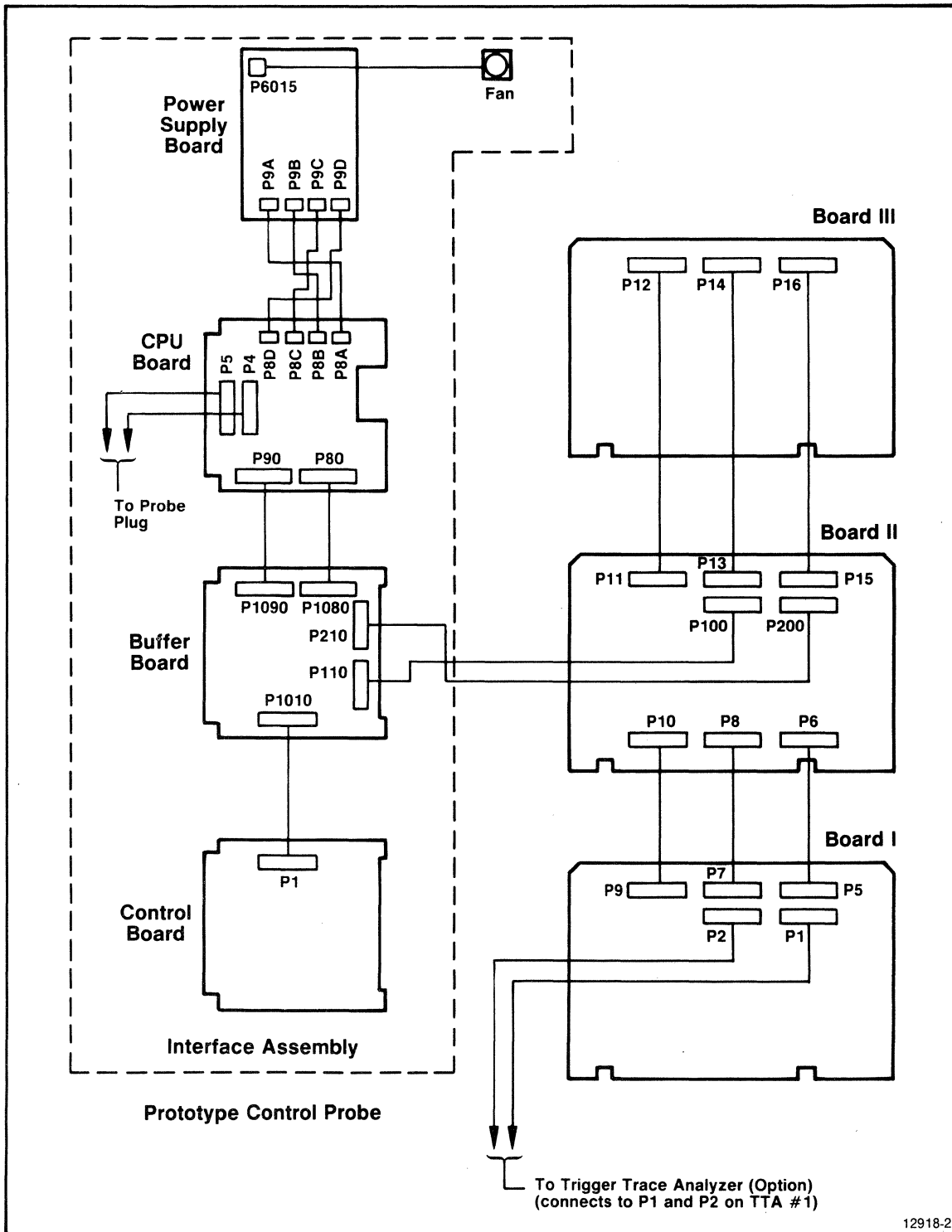


Figure 6-7. Emulator boards with prototype control probe.

Connecting to the Prototype

CAUTION

Static discharge can damage the prototype control probe when the probe plug is not installed in a 68-pin Textool socket. When handling the probe plug, hold the probe plug by the case only and do not touch the pins. The probe plug should be stored in conductive foam when not in use.

The 68-pin JEDEC Type A probe plug at the end of the four 1-foot flexible cables fits into a 3M Textool 68-pin chip carrier socket in the prototype hardware. Pin 1 on the plug must be mated with the corresponding pin 1 in the socket. A mark is located near pin 1 on both the plug and the socket to aid in pin identification, as illustrated in Figure 6-8.

CAUTION

If the prototype control probe plug is incorrectly inserted in the prototype socket, damage to the prototype control probe or to the prototype may result. Refer to Figure 6-8 and the following procedure to ensure proper plug insertion.

Probe Plug Insertion Procedure

To ensure that the probe plug is properly inserted into the 3M Textool socket, follow these steps:

NOTE

Three of the probe plug's corners contain square notches that are used as indexes to ensure correct plug alignment. The fourth corner, Pin No. 1 Mark, is cut on a 45 degree bias. (Refer to Figure 6-8.) The 3M Textool socket has three guide boss pins that mate with the square notches and a spring lever that mates with the corner marked as Pin No. 1.

1. Before inserting the probe plug, inspect the socket to verify the positions of the contact fingers. If the socket is bent, damaged, or has contaminated contact fingers, the socket should be replaced with a new socket before installing the probe.
2. Move the wire bail on the socket to its open position. Ensure that the slide fasteners on the probe plug are in their retracted positions. Figure 6-8 shows the slide fasteners in their retracted positions.
3. Carefully lower the probe plug, perpendicular to the plane of the socket, into the socket. Do not slide or twist the plug into the socket. These movements may damage or bend the contact fingers. The two stationary tabs on the plug should be toward the wire bail on the socket.
4. Move the probe plug around very gently to verify that the square notches on the plug are aligned with the guide boss pins in the socket. When the plug is properly seated and not hung up on one or more of the guide boss pins, push the probe plug back toward the corner opposite the corner marked Pin No. 1. This ensures that the contact pads are aligned properly.

5. While holding the probe plug in this position, press down on the plug case until the stationary tabs are low enough to pull the wire bail over them, keeping that end of the plug in place.
6. While still maintaining pressure on the plug, move the slide fasteners on each side of the plug to their fully extended position so that the tabs on the fasteners fit into the holes on the raised back corners of the socket. When the wire bail and slide fasteners are in place, remove pressure from the plug case.

CAUTION

Be very careful when handling the flexible cable attached to the probe plug. If the cable is creased or nicked at the edges, excessive pulling on the cable could cause a tear. Tears can propagate rapidly through the circuit runs, causing open and shorted circuits.

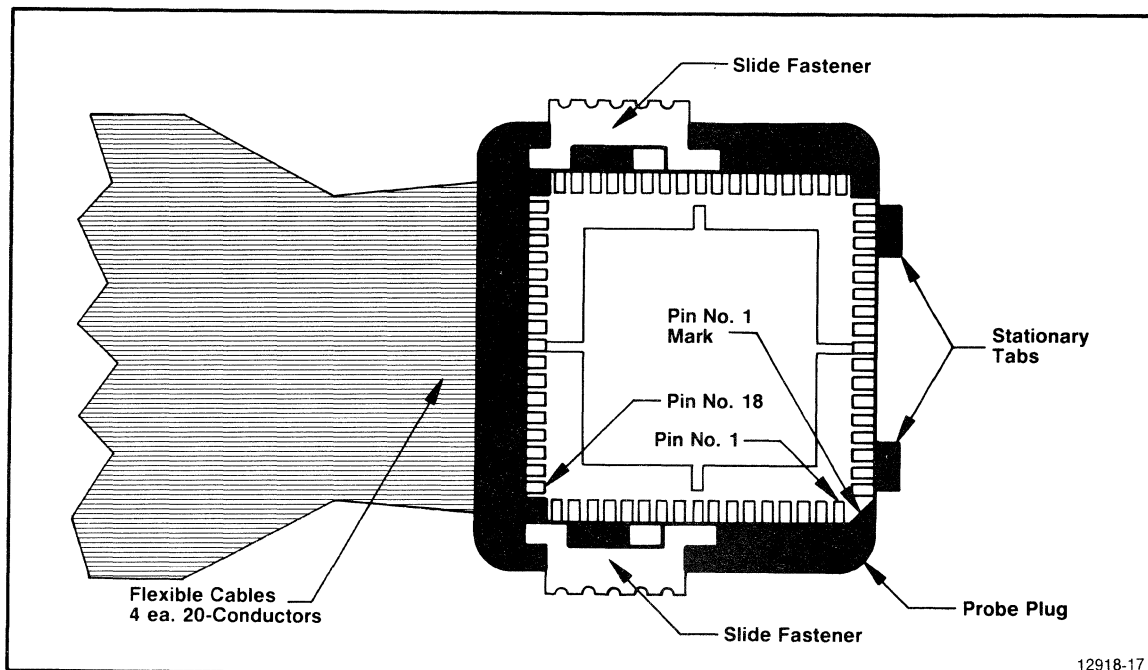


Figure 6-8. Pin identification and proper plug insertion.

Prototype Control Probe Disassembly Procedure

The following procedure tells you how to disassemble the prototype control probe. This procedure should be followed whenever changing the jumpers on any of the boards or calibrating the power supplies. The prototype control probe consists of an Interface Assembly which houses the four circuit boards and fan. Two 6-foot, 40-conductor ribbon cables (connecting to P110 and P210 on the Buffer Board) are the interfacing cables to the emulator boards. Four 1-foot, 20-conductor flexible cables connecting to P4 and P5 on the CPU Board are the interfacing cables to the probe plug.

The Interface Assembly housing consists of a top cover, bottom cover, and spacer ring (the spacer ring forms the sides of the housing). The disassembly procedures are presented in three parts depending on how far you want to disassemble the probe:

- Interface Assembly Disassembly
- Top Cover Disassembly
- Bottom Cover Disassembly

Interface Assembly Disassembly Procedure

1. Ensure that the primary power (115 Vac or 230 Vac) to the development system is OFF.
2. Place the prototype control probe, top cover down, on a flat nonconductive working surface with the flexible cables and probe plug facing you.
3. Remove the four screws from the bottom cover of the Interface Assembly's housing (one screw in each corner of the bottom cover).
4. Grasp the Interface Assembly's housing with both hands (holding both top and bottom of the housing together) and turn the assembly onto its bottom.
5. Lift up carefully on the top cover and spacer ring, while rotating the cover and ring to the left (counterclockwise) 180 degrees. Set the top cover and spacer ring beside the bottom cover (the top cover is now laying to the left of the bottom cover).

NOTE

The CPU Board and Power Supply Board are attached to the top cover. The Buffer Board and Control Board are attached to the bottom cover. With the top and bottom covers laying side by side, the CPU Board and the Buffer Board are exposed, and the jumpers for both boards are readily accessible. If you want to calibrate the power supplies, follow the procedure for "Top Cover Disassembly Procedure." If you want to change the jumpers on the Control Board, follow the procedure for "Bottom Cover Disassembly Procedure."

Top Cover Disassembly Procedure

1. Using a 1/4-inch nut driver, remove the six nuts and lock washers attaching the CPU Board to the spacer studs.

NOTE

The following procedure may require more than one person to prevent damage to the circuit boards and interconnecting cables.

2. Lift the CPU Board carefully off the spacer studs, lifting the bottom cover at the same time. Rotate both the CPU Board and bottom cover to the left (counterclockwise) 180 degrees and lay them beside the top cover. (The CPU Board is now laying to the left of the top cover and the bottom cover is to the left of the CPU Board.)

NOTE

The Buffer Board (attached to the bottom cover) is connected to the CPU Board with two ribbon cables. The Power Supply Board is connected to the CPU Board with four ribbon cables. Take care not to disconnect or damage these cables when moving the bottom cover and the CPU Board.

3. Remove the six spacer studs from the shield covering. Remove the metal shield and set it aside.

The Power Supply Board is now accessible, and the power supplies may be calibrated. Refer to "Power Supply Calibration" in Section 4 of this manual.

Bottom Cover Disassembly Procedure

NOTE

This procedure assumes that the top and bottom covers are laying side by side (top cover is to the left of the bottom cover).

1. Using a 1/4-inch nut driver, remove the six nuts and lock washers attaching the Buffer Board to the spacer studs.
2. Remove the ribbon cable connector P1010 from the Buffer Board.
3. Lift the Buffer Board carefully from the spacer studs. While lifting the Buffer Board, at the same time move the bottom cover to the right. Lay the Buffer Board between the top and bottom covers.
4. Using a 1/4 inch nut driver, remove the six spacer studs from the back side of the Control Board.
5. Remove the Control Board board and turn its component side upwards. The jumpers are now accessible.
6. Reassemble the Interface Assembly in reverse order of disassembly.

Connecting to the Trigger Trace Analyzer (Optional)

The following procedure explains how to connect the 80186/80188 Emulator to the optional Trigger Trace Analyzer (TTA). Refer to the *Trigger Trace Analyzer Installation Manual* for further information.

Figures 6-7 and 6-9 shows the connections between the TTA and the 80186/80188 Emulator. To install the TTA with the emulator, follow these steps:

1. Install the 80186/80188 Emulator in your development system using the procedures described earlier in this section. Don't replace the cover of the development system until this entire procedure is completed.
2. On the back side of TTA Board No. 1, attach one of the emulator interconnecting cables to P1 and the other cable to P2.
3. Install the TTA, using the procedures described in the *Trigger Trace Analyzer Installation Manual*.
4. Attach the two interconnect cables from the TTA to P1 and P2 of Emulator Board I.
5. Replace the top cover of the development system using the procedures described earlier in this section.

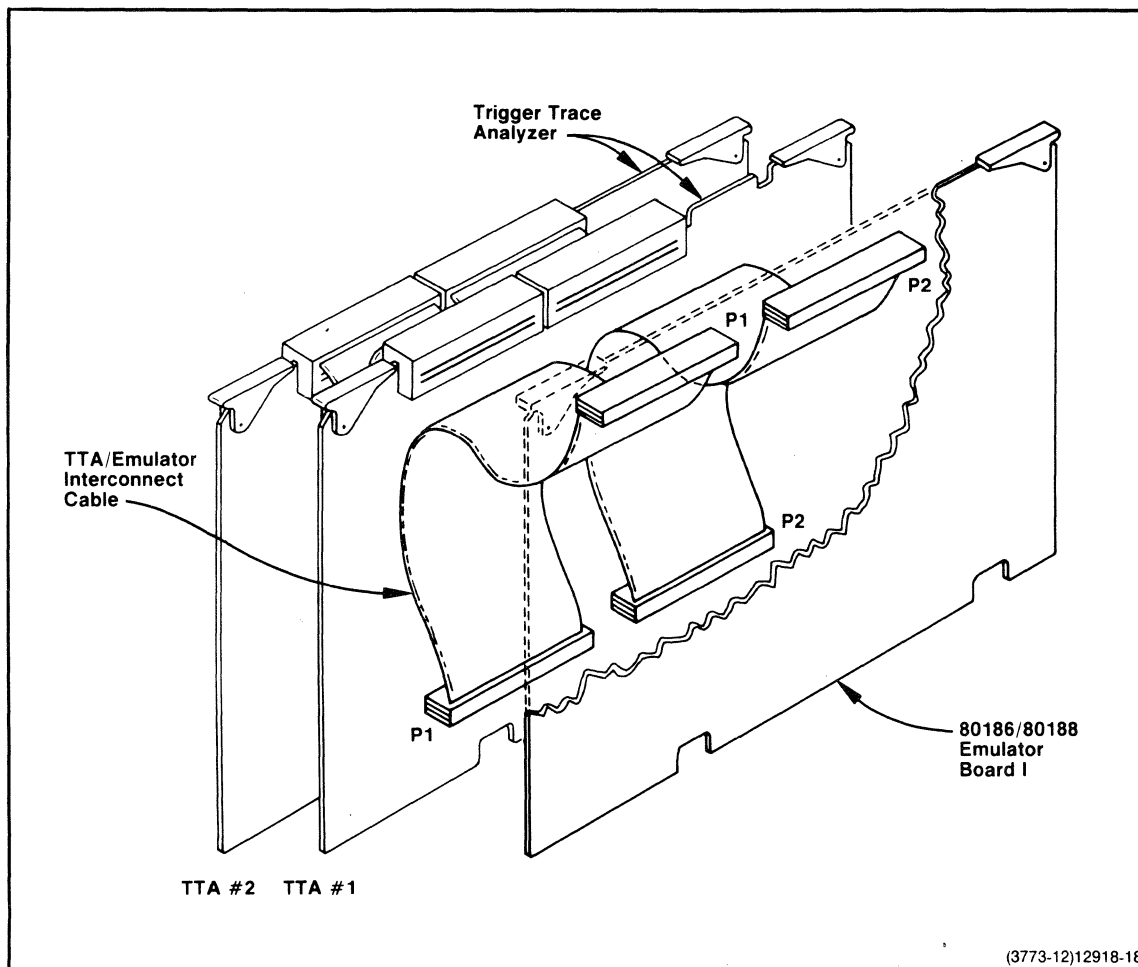


Figure 6-9. 80186/80188 Emulator Board I connections to the TTA.

(3773-12)12918-18

Grounding

A proper ground system is necessary for the satisfactory operation of the 8301 or the 8540. The emulator boards, prototype control probe, other optional equipment, and peripheral equipment must be properly grounded to eliminate ground loops and prevent static discharge damage. Ensure that primary power cords of all units, including the prototype circuitry, are connected to outlets on the same ground system.

Refer to your development system Installation Guide for additional grounding procedures.

SOFTWARE INSTALLATION

Software installation procedures are provided in the following pages. These procedures consist of:

- Setting up the 8560 Multi-User Software Development Unit
- Installing the firmware for an 8540 Integration Unit
- Installing the software for an 8550 Microcomputer Development Lab

Setting up the 8560

The following text describes procedures that you must perform if you are using an 8560 with your 8540 or 8550.

Setting Your Shell Variable

If you are using an 8540 or an 8550 in the TERM mode with an 8560, and want to assemble 80186/80188 code, you must set your shell variable to the assembler you are going to use. To use the 80186/80188 Assembler (ASM80186 version x.xx), you must set your shell variable as follows each time you login:

```
$ uP=80186; export uP
```

Installing the 8540 Firmware

The firmware shipped with your emulator contains emulator control and diagnostic ROMs. These ROMs are installed in your 8540's System ROM Board. The firmware package consists of one type-2764 ROM and four type-27128 ROMs:

- One type-2764 ROM labeled "BASE DIAG 1" is a replacement for one of the diagnostic ROMs presently installed in your 8540 System ROM Board.
- Three type-27128 ROMs labeled "80186/188-1", "80186/188-2", and "80186/188-3" contain the emulator control firmware.
- One type-27128 ROM labeled "80186/188 DIAGNOSTICS" contains the emulator diagnostics.

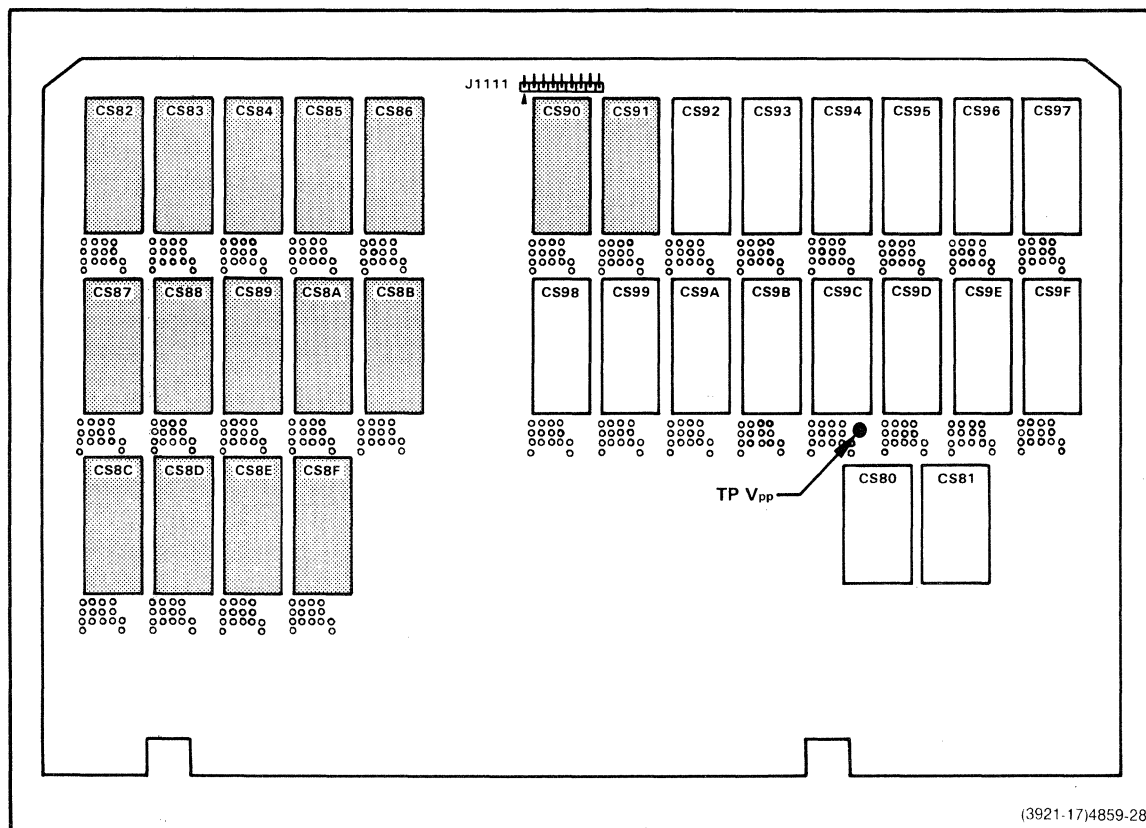


Figure 6-10. System ROM Board socket locations.

The following procedure explains how to perform this installation.

1. Remove the 8540's top cover by completing steps 1 through 3 described earlier in this section under "Installing the Emulator Boards and Prototype Control Probe."
2. Remove the System ROM Board from your system's Main Interconnect Board and place the board on a flat static-free work area.
3. Check the part number on the ROM labeled "BASE DIAG 1" installed in your System ROM Board. BASE DIAG 1 is normally installed in the socket labeled CS94 shown in Figure 6-10. The socket circuit designator is U1160. If the BASE DIAG 1 part number (160-1378-xx) is lower than the part number on the BASE DIAG 1 ROM shipped with your emulator, then replace the installed ROM with the new ROM. (Install the BASE DIAG 1 ROM with the highest extension number.)
4. The four type-27128 ROMs shipped with your emulator may be installed in any available sockets you choose. However, it is recommended that emulator option ROMs be installed in any of the ten sockets labeled CS82–CS8B shown in Figure 6-10. (The socket circuit designators are U1010, U1030, U1050, U1060, U1070, U3010, U3030, U3050, U3060, and U3070).

NOTE

The spare ROM sockets for options are limited. It may be necessary to remove un-needed ROMs to install the required 80186/80188 Emulator ROMs. (No other emulator can be installed in the 8540 with the 80186/80188 Emulator.)

NOTE

The 8540's System ROM Board was originally shipped from the factory with the ROM sockets configured for only type-2764 ROMs. In recent shipments of the System ROM Board, the ROM sockets CS82 through CS91 are configured for both type-2764 and type-27128 ROMs. See Figure 6-11. If your 8540's System ROM Board is not configured for type-27128 ROMs, sixteen 0-ohm resistors are provided with the 80186/80188 Emulator to make this modification to ROM sockets CS82 through CS91 before the type-27128 ROMs are installed.

5. Solder a 0-ohm resistor (provided with your emulator) across the lower strap positions of each socket CS82 through CS91. The sockets CS82 through CS91 are shown as shaded areas in Figure 6-10. Figure 6-11 shows the position of the 0-ohm resistor. After a socket is modified, either a type-2764 or a type-27128 ROM can be installed in the socket. Insert the four type-27128 ROMs provided with your emulator into any four sockets CS82 through CS8B shown in Figure 6-10.

NOTE

If the 0-ohm resistor is missing or damaged, you may substitute a solid wire strap.

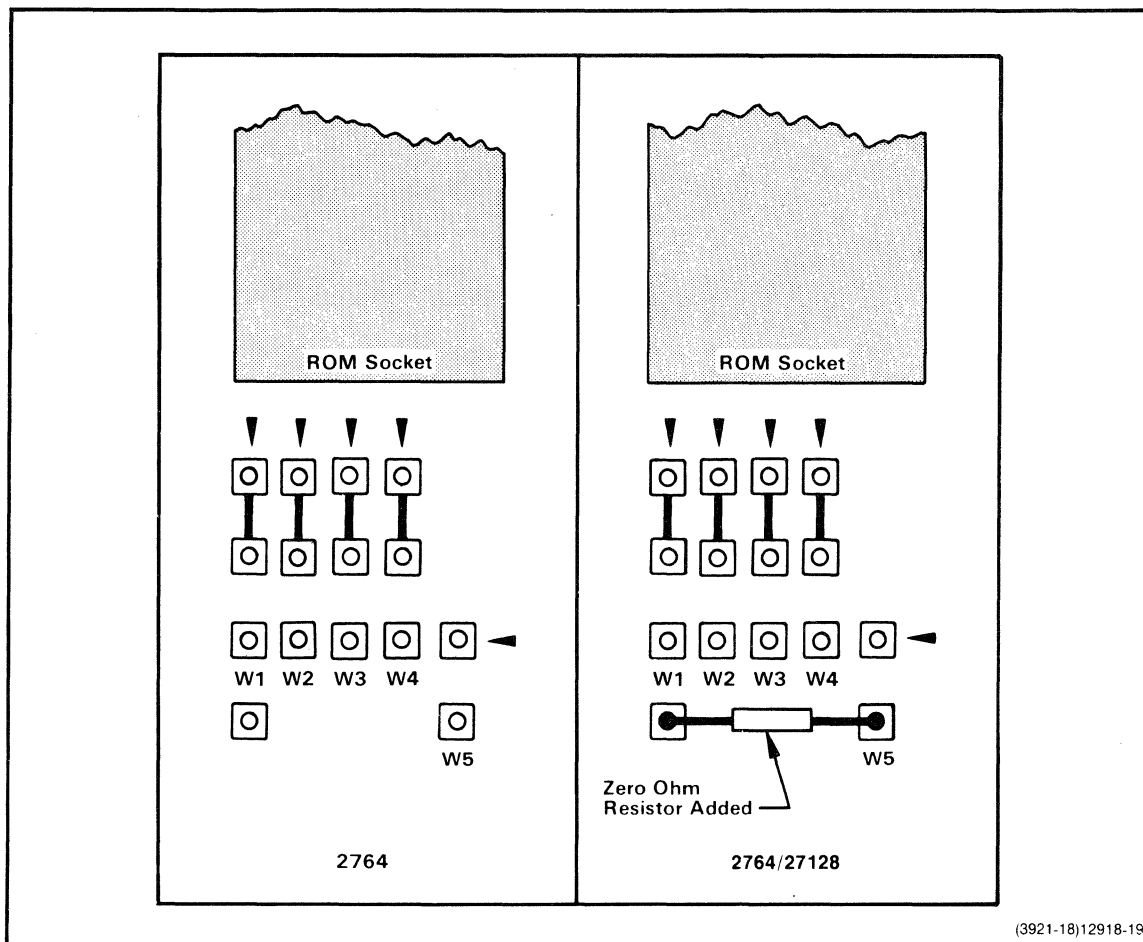


Figure 6-11. Type-2764 and type-27128 ROM strapping.

6. Install other optional ROMs on the System ROM Board as follows:
 - Install the PROM Programmer option ROMs in locations CS90 and CS91 shown in Figure 6-10. The socket circuit designators are U1110 and U1120.
 - Install the Trigger Trace Analyzer option ROMs in locations CS8C, CS8D, and CS8E shown in Figure 6-10. The socket circuit designators are U4010, U4030, and U4050.
 - Install the Communications option (COMM Option) ROM in location CS8F shown in Figure 6-10. The socket circuit designator is U4060.
7. Reinstall the System ROM Board in the mainframe exactly as it was before. Replace the top cover by completing steps 14 and 15 as described earlier in this section under "Installing the Emulator Boards and Prototype Control Probe."

Installing the 8550 Software

The following text explains how to install the flexible disks shipped with your emulator. This discussion also explains how to use the 8086/8088 Assembler to assemble 8086 and 80186 code providing the code does not contain instructions unique to the 80186 or 80188 microprocessors.

NOTE

The 8550 development system does not have an 80186/80188 Assembler. If you have an 8086/8088 Assembler for the 8550, the control software provided with your 8086/8088 Emulator can be installed onto your DOS/50 system disk. Installing this software permits you to assemble and load an 8086 program or 80186 program providing the code does not contain the unique instructions for the 80186 or 80188 microprocessors. The 8086/8088 Assembler is not compatible with instructions that are unique to 80186 or 80188 microprocessors.

Your emulator software package for an 8550 installation consists of two disks:

- A disk containing the emulator control software, which you install onto your DOS/50 system disk so that DOS/50 can control your emulator hardware.
- An 8550 System Diagnostic Disk, which contains the latest version of your development system's diagnostic program.

The following paragraphs describe how to install the control software for your 80186/80188 Emulator.

To complete this installation procedure you need the following items:

- An 8550 system (with or without an 80186/80188 Emulator)
- A DOS/50 system disk with a write-enable tab over the write-protect slot
- An 80186/80188 Emulator Software Installation Disk without the write-enable tab

Start Up and Set the Date

Turn on your 8550 system. (For start-up instructions, refer to the Learning Guide of your System Users Manual.) Place your system disk in drive 0 and shut the drive 0 door. When you see the > prompt on your system terminal, place your installation disk in drive 1 and shut the drive 1 door.

Use the **dat** command to set the date and time. For example, if it is 11:05 a.m. on October 12, 1984, enter:

```
> dat 12-oct-84/11:05
```

The system uses this information when it sets the CREATION time attribute of each file copied from your installation disk.

Installation Procedure

The command file *INSTALL2*, which installs the emulator control software, resides on the installation disk. To execute the command file, simply type its filespec:

```
> /VOL/EMU.80186/INSTALL2
```

DOS/50 responds with the following message:

```
* During this installation procedure, one or more of the
* following messages may appear.  IGNORE THESE MESSAGES:
*
*      Error 6E - Directory alteration invalid
*      Error 7E - Error in command execution
*      Error 1D - File not found
*
* If any OTHER error message appears, see your
* Users Manual for further instructions.
*
* If no other error message appears, you'll receive a
* message when the installation procedure is complete.
*
T,OFF
```

In this installation procedure, you may disregard error messages 6E, 7E, and 1D; these messages have no bearing on the success of the installation. However, if a message other than 6E, 7E, or 1D appears, take the following steps:

1. Make sure you are using the right disks.
2. Make sure your system disk has a write-enable tab.
3. Make sure there are at least 16 free files and 150 free blocks on your system disk.
4. Begin the installation procedure again.

If the installation procedure fails again, copy down the error message and contact your Tektronix service representative.

The **T,OFF** command in the preceding display suppresses subsequent output to your system terminal (except error messages) until *INSTALL* finishes executing. Within about five minutes, *INSTALL* will finish and your system terminal will display the following message:

```
*
* Your installation has been completed.
>
```

Once your software is installed, you can:

- Remove your disks and turn off your 8550 system
- Install more software onto your system disk
- Continue with the 80186/80188 Emulator demonstration run provided in Section 3 of this manual if your 80186/80188 Emulator is installed. If you continue, you do not have to restart the system or reset the date and time.

NOTE

At this point, "no.name" is the current user. To change the current user back to "yourname," enter:

```
> user,,yourname
```

Installing the Diagnostic Software

The Diagnostic Software Disk provided with your 80186/80188 Emulator replaces the Diagnostic Software Disk provided with your development system. Therefore, you may discard or write over your original disk. For more information on how to run the diagnostic software, refer to Section 7 of this manual or to the *80186/80188 Emulator Service Manual*.

Using Your 8086/8088 Assembler on the 8550

If you are using an 8550 and want to assemble 80186/80188 code (providing the code does not contain instructions unique to the 80186 or 80188 microprocessors), you must use the 8086/8088 Assembler. To use the 8086/8088 Assembler, enter the following commands one time only. Your 8086/8088 Assembler and 80186/80188 Emulator Control Software must be installed and the user set to TEKTRONIX.

```
> fl /EOS/8086/ASM[] /EOS/8086/ASM[]

Flink ASM[]                to ASM[]      ?y

> fl /EOS/8086/ASM.3 /EOS/8086/ASM.3

Flink ASM.3                to ASM.3      ?y
```

Section 7

VERIFICATION PROCEDURES

INTRODUCTION

This section contains information about performing operational test procedures and functional verification procedures for the 80186/80188 Emulator. Operational test and verification procedures are given for both an 8550 and an 8540.

This section is divided into three parts. The first part discusses the equipment and test procedures necessary to verify emulator operation. The second part discusses the equipment necessary to verify timing relationships of signals at the prototype control probe's probe plug pins. The last part describes how to perform a system functional verification.

Every time you install an emulator in your development system, you should run the automatic system verification checks. Emulator diagnostics are run automatically as part of the system diagnostic test program. Procedures for running this diagnostic test program are included in this section and in your development system's Installation Guide.

OPERATIONAL TEST PROCEDURE

The operational test procedures tell you how to verify the emulator's operation using the MicroLab I with the 80186/80188 Personality Card.

Equipment Required

This equipment is required to check the operation of the installed 80186/80188 Emulator:

- TEKTRONIX MicroLab I (067-0892-xx)
- 80186/80188 Personality Card (018-0211-00)

The MicroLab I checks the 80186/80188 Emulator by providing a circuit with known characteristics (the personality card). The personality card is monitored by MicroLab I circuitry, and test results are indicated on the MicroLab I display. The MicroLab I operating system also contains tests that exercise the functions of the prototype control probe.

Throughout this section it is assumed that you are familiar with the MicroLab I and its characteristics. For more information about this equipment, refer to the *MicroLab I Instruction Manual* and the *80186/80188 Personality Card Supplement*.

Equipment Setup

1. Ensure that the power to the development system and to the MicroLab I is turned off.
2. Verify that all jumpers are installed correctly on the personality card. (Refer to the 80186/80188 Personality Card Supplement.)
3. Install the personality card in the MicroLab I.
4. Insert the 68-pin probe plug of the prototype control probe in the Textool 68-pin chip carrier socket on the 80186/80188 Personality Card. For proper probe plug insertion, refer to "Connecting to the Prototype" in Section 6 of this manual.

Functional Test Procedure

1. Turn on the power to the development system and to the MicroLab I.
2. Start up the operating system. For detailed information about system operation, refer to your System Users Manual.
3. Enter the following command to identify the emulator to be tested. If you are using an 8550, enter:

```
> sel 80186          or          > sel 80188
```

If you are using an 8540, enter:

```
> sel 80186
```

4. Enter the desired emulation mode. Perform step a **or** b:
 - a. To verify memory mapping capability, select emulation mode 1, by entering:

```
> em 1
```

When the prompt character `>` is displayed, enter:

```
> map u 00000 0FFFFF
```

This transfers all memory to the MicroLab I.
 - b. If only the operation of the emulator is to be tested, select emulation mode 2, by entering:

```
> em 2
```

5. Enter the following command to start program execution:

```
> reset
```

Then, while holding down the RESET key on the MicroLab I, enter:

```
> g 0
```

Release the RESET key.

If the 80186/80188 Emulator is operating properly, the MicroLab I will display "HELLO". This display indicates that most of the emulator circuitry is working properly. However, several control lines are not checked during initialization and should be verified with the MicroLab I processor tests, explained later in this section.

No HELLO Display

If the MicroLab I does not display “HELLO”, a problem exists with the 80186/80188 Emulator boards, the prototype control probe, or the MicroLab I. Use the following procedure to determine which unit is not working:

1. Turn off the power to the MicroLab I and to the development system.
2. Disconnect the probe plug from the personality card's 68-pin chip carrier socket.

CAUTION

The 80186 or 80188 microprocessor can be damaged by static discharge when not installed in a socket. Use extreme caution and observe anti-static precautions when handling the leadless chip carrier. Do not touch the pins. The microprocessor should be stored in conductive foam when not in use.

3. Install the personality card's 80186/80188 microprocessor device in the 68-pin chip carrier socket.
4. Turn on the power to the MicroLab I.

If the MicroLab I now displays “HELLO”, the problem is with the 80186/80188 Emulator or the prototype control probe. For information on further troubleshooting of this problem, refer to the diagnostics discussed later in this section.

If the MicroLab I does not display “HELLO”, a problem exists with the MicroLab I. Refer to the *MicroLab I Instruction Manual* and *80186/80188 Personality Card Supplement* for servicing information.

Processor Test Procedure

If the emulator has passed the functional test procedure, you are ready to run the processor tests. While each test is being performed, the display will show “PROC n” (n is the number of the test being performed).

Perform the following procedure:

1. Press the RESET key on the MicroLab I keypad. This initializes the processor test hardware in the MicroLab I.
2. Press the PROC TEST (Shift 1) key to start the processor tests. The display will show “Pn”.
3. Press the 0 key. The MicroLab I will execute the PROC 0 test and then display “SPECIAL”.
4. Press the SPECIAL key. The MicroLab I performs the tests PROC 1 through PROC 4 and part of PROC 5 test automatically, and then displays “SPECIAL” again.

NOTE

Test PROC 1 is executed if jumper P1091 on the personality card is in the Queue Status mode. If P1091 is in the Normal mode, PROC 1 is skipped and PROC 2 test is executed.

5. Press the SPECIAL key again to complete PROC 5 test and automatically perform the tests PROC 6 through PROC 8. When PROC 8 is finished, the MicroLab I displays "rEAdY".

NOTE

Processor tests PROC 9, PROC A, PROC B, and PROC C are stand-alone tests. Only one test is conducted depending on the interrupt configuration of your emulator.

6. Table 7-1 lists the four stand-alone processor tests and shows the Interrupt Controller configuration mode for each test. Press only one key 9, A, B, or C depending on which test you want. When this test is finished, the MicroLab I again displays "rEAdY".

**Table 7-1
Stand-Alone Processor Tests**

Processor Test	Interrupt Controller Configuration Mode
PROC 9	Fully Nested (Direct) Mode
PROC A	Direct/Cascade Mode
PROC B	Cascade/Direct Mode
PROC C	Cascade Mode

Refer to the *80186/80188 Personality Card Supplement* and to the *MicroLab I Instruction Manual* for processor test error codes.

Successful completion of both the functional test and the processor test procedures verifies that the 80186/80188 Emulator is operational.

EMULATOR TIMING VERIFICATION

Occasionally you may want to verify timing relationships between signals at the pins of the prototype control probe's probe plug. The following paragraphs discuss equipment necessary to perform these timing verifications.

Section 4 of this manual provides timing diagrams which show timing relationships.

Measurement Considerations

Emulator timing verification involves measurement of extremely fast signals. Test equipment used for these measurements should resolve timing differences of less than 5 ns between two signals. A resolution of 1 ns is best for examining the most critical timings.

Be careful that the test equipment does not introduce errors. Check the test equipment calibration carefully. If you are using a dual trace oscilloscope rather than a dual beam model, be sure to account for any possible skew between the two input channels. In general, good laboratory measurement practices ensure accurate results.

Equipment Required

To measure timing relationships with the preferred accuracy and resolution, you may use the following equipment:

- A TEKTRONIX 7844 Dual Beam Oscilloscope, or equivalent
- Two TEKTRONIX 7A26 Vertical Amplifiers, or equivalent
- A TEKTRONIX 7B85 Delaying Time Base, or equivalent

Controlling the Signal Lines Under Test

Some processor signal lines, such as interrupt lines, are normally connected to asynchronous circuits. Timing relationships of asynchronous signals may be difficult to measure with an oscilloscope. To exercise these signal lines in a periodic manner, you may have to develop software routines or use an external test fixture such as the TEKTRONIX MicroLab I.

SYSTEM FUNCTIONAL VERIFICATION

The system functional verification procedures tell you how to verify the operation of the 80186/80188 Emulator using your development system's automatic system verification diagnostics.

To perform verification of the 8550, you'll need the system terminal connected to the 8301, and the 8550 System Diagnostics Disk in the 8501's disk drive.

To perform verification of the 8540, you'll need the system terminal connected to the 8540, and the 80186/80188 Diagnostics ROM installed in the 8540's System ROM Board.

This section does not provide a detailed description of the 8550 disk-resident diagnostics or the 8540 ROM-resident diagnostics. The information provided here gives you the quickest procedure to verify the 80186/80188 Emulator's operation.

For detailed information about your development system's disk-resident or ROM-resident diagnostics, refer to the following optional service manuals:

- *8301 Microprocessor Development Unit Service Manual*
- *8540 Integration Unit Service Manual*

For more information about the emulator-specific diagnostics, refer to the *80186/80188 Emulator Service Manual*.

8550 Microcomputer Development Lab Verification

When both the 8501 and 8301 have displayed the boot messages on the system terminal, you are ready to run the 8550 disk-resident diagnostics.

Procedure

To verify 80186/80188 Emulator operation, perform the following procedure:

1. Insert the 8550 System Diagnostics Disk label side up into the 8501's drive 0 (top drive).
2. Close the disk-drive door.

Within 6 seconds, the 8501 will begin a preliminary read of the disk and the system terminal will display the following information:

```
8301 BOOT V x.x
8501 V x.x
8301 BOOT V x.x
```

3. The 8501 will begin reading the disk again.

Approximately 12 seconds later, the diagnostic greeting message and first menu will appear on the system terminal. The message and menu are shown in Display 7-1.

```
*****
*           TEKTRONIX INC.           *
*      8550 DISK-RESIDENT DIAGNOSTIC SYSTEM      *
*      VERSION X.X                      *
*      Copyright (C) 1984 Tektronix, Inc.        *
*****

                RUN MODE MENU
                -----

1 - AUTOMATIC MODE    *** default ***
2 - SELECT MODE
H - HELP

Type in desired mode    {<CR> or 1, 2, or H and <CR>}
```

Display 7-1.

4. Press the RETURN key to select the Automatic Mode (system verification) as soon as the system terminal displays the diagnostic greeting message and Run Mode Menu. Immediately, the Automatic Mode Menu is displayed.
5. Press the RETURN key again to select the automatic system verification tests. The Display Option Menu is immediately displayed.
6. Press the RETURN key a third time to select terminal display and to start execution of the automatic system verification tests. You will not need to intervene further.

Various test running messages for both the 8301 and 8501 are displayed while the verification tests are executing. The 8301 (including the 80186/80188 Emulator) is tested first, followed by the 8501. The diagnostic tests take approximately 10 minutes to execute. At that time, the system terminal will display one of these messages:

SYSTEM VERIFICATION PASSED

or

SYSTEM VERIFICATION FAILED

If the SYSTEM VERIFICATION FAILED message is displayed, refer to the *8301 Microprocessor Development Unit Service Manual* and the *8501 Data Management Unit Service Manual* for information about performing exhaustive diagnostic troubleshooting.

This completes the functional test procedure for an 80186/80188 Emulator installed in an 8550.

8540 Integration Unit Verification

Procedure

NOTE

The 80186/80188 Diagnostic ROM must be installed in your 8540's System ROM Board before you can verify the emulator's operation. If not already installed, refer to Section 6 of this manual for ROM installation procedures.

1. Check that the Mode Selector switch on the 8540's System Controller Board is in its normal operating configuration: switch positions 0 through 2 and 4 through 7 should be set to 0, switch position 3 should be set to 1.
2. Power up the 8540.

After the 8540 has displayed its boot message on the system terminal, you're ready to run the 8540 ROM-based diagnostics.

3. Enter the following command at the system terminal:

```
> sel diags
```

The system terminal will display the diagnostic greeting and the Run Mode Menu, as shown in Display 7-2.

```
*****
*   TEKTRONIX INC.   *
*   8540 ROM-RESIDENT DIAGNOSTIC SYSTEM   *
*   VERSION X.X      *
*   Copyright (C) 1984 Tektronix, Inc.     *
*****
```

RUN MODE MENU

1 - AUTOMATIC MODE *** default ***
2 - SELECT MODE
H - HELP

Type in desired mode {<CR> or 1, 2, or H and <CR>}

Display 7-2.

4. Press the RETURN key to select the Automatic Mode (system verification) as soon as the system terminal displays the diagnostic greeting message and Run Mode Menu. Immediately, the Automatic Mode Menu is displayed.
5. Press the RETURN key again to select the automatic system verification tests. The Display Option Menu is immediately displayed.
6. Press the RETURN key a third time to select terminal display and to start execution of the automatic system verification tests. You will not need to intervene further.

Various test running messages are displayed while the verification tests are executing. The diagnostic tests take a approximately 10 minutes to execute. At that time, the system terminal will display one of these messages:

SYSTEM VERIFICATION PASSED

or

SYSTEM VERIFICATION FAILED

If the SYSTEM VERIFICATION FAILED message is displayed, refer to the *8540 Integration Unit Service Manual* for information about performing exhaustive diagnostic troubleshooting.

This completes the functional test procedure for an 80186/80188 Emulator installed in an 8540.

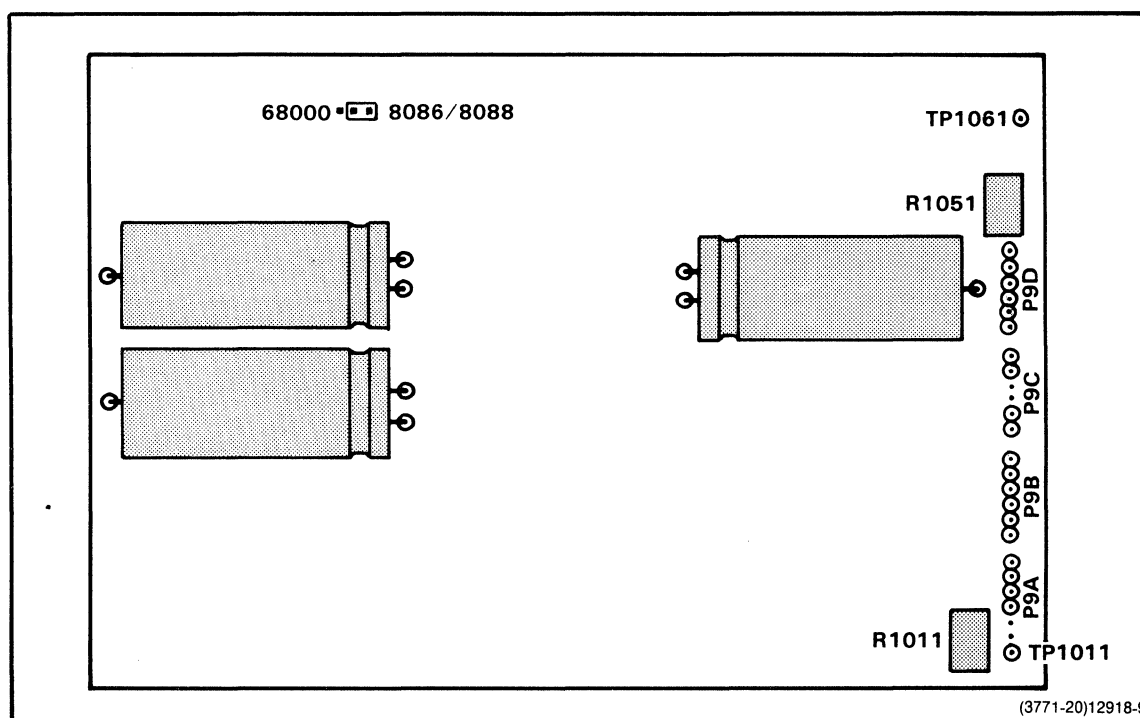


Figure 4-6. Probe Power Supply Board.

SPECIFICATIONS

This paragraph lists the electrical, environmental and physical characteristics for the 80186/80188 Emulator.

Electrical Characteristics

Table 4-2 lists the electrical characteristics for the 80186/80188 Emulator.

Table 4-2
Electrical Characteristics

Characteristics	Performance Requirement	Supplemental Information
Supply Voltage	+5.2 Vdc +1%/-2% +12.0 Vdc $\pm$ 5% -12.0 Vdc $\pm$ 5%	-12.0 Vdc used by prototype control probe only
Current (typical)		+5.2 Vdc @ 7.00 A +12.0 Vdc @ 1.25 A -12.0 Vdc @ 0.40 A
Power Dissipation		55 W (max.) selected 40 W (max.) not selected
Buffer Board Fuses F9035 F9050 F9055		3AG, 1.5 A, 250 Vdc, fast 3AG, 0.5 A, 250 Vdc, fast 3AG, 3.0 A, 250 Vdc, fast

Environmental Characteristics

Table 4-3 lists the environmental characteristics for the 80186/80188 Emulator.

Table 4-3
Environmental Characteristics

Characteristics	Description
Temperature Operating	0°C to +50°C (32°F to 122°F)
Storage	-55°C to +75°C (-67°F to +167°F)
Relative Humidity	90% maximum non-condensing
Altitude Operating	5 000 m (15,000 ft) maximum
Storage	16 400 m (50,000 ft) maximum

Physical Characteristics

Table 4-4 lists the physical characteristics for the 80186/80188 Emulator.

Table 4-4
Emulator Physical Characteristics

Characteristics	Height	Length	Width
Emulator Boards Board I, Board II, and Board III	195 mm (7.68 in)	280 mm (11.0 in)	
Prototype Control Probe Interface Assembly	102 mm (4.0 in)	236 mm (9.25 in)	185 mm (7.25 in)
Power Supply, Control, Buffer, and CPU boards	166 mm (6.5 in)	107 mm (4.2 in)	